Learning to Act: Acquisition and Optimization of Procedural Skill

Frank J. Lee
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15232-3890
FJL@CMU.EDU

John R. Anderson
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15232-3890
JA@CMU.EDU

Abstract

People become highly reactive when performing dynamic, real-time tasks like driving a car, playing a video game, or controlling air traffic. However, people also go through more deliberate stages in which they spend time reasoning about constraints and actions. In this paper, we argue that these are the end points on a learning continuum, and we discuss one potential mechanism for bridging these endpoints within the ACT-R (Anderson, 1993) framework.

Keywords: Procedural skill optimization, Learning in dynamic task environments, Cognitive models of problem solving.

Introduction

Performing dynamic tasks requires quick reaction to an ever-changing, dynamic environment. While driving a car, for instance, we have neither the time nor the luxury to perform deep-level planning and problem solving when another car suddenly veers into our lane. Because of the narrow slice of time within which people must perceive their situation and act, some researchers (e.g. Agre & Chapman, 1988) have argued against deliberative problem solving methods like means-ends analysis (Newell & Simon, 1972) in dynamic task domains. They claim that the time required for classical planning methods like means-ends analysis is simply too long for dynamic tasks, in which one must react quickly. Yet it is also the case that when people are learning to perform novel tasks, they use deliberative problem solving methods like means-ends planning (Anderson, 1982; Newell & Simon, 1972). This raises the following question: How does this transition from deliberative to reactive problem solving occur?

We propose a potential mechanism that underlie this transition within the ACT-R (Anderson, 1993) framework using the Kanfer-Ackerman Air-Traffic Controller©[1] (ATC) Task. The Kanfer-Ackerman ATC Task is useful, because Ackerman (1994) has collected data from over 3500 subjects on the Kanfer-Ackerman ATC Task and has made them available on a CD-ROM (Ackerman & Kanfer, 1994) to the Office of Naval Research (ONR).

In this paper, we first outline a process by which task instructions (encoded as declarative knowledge) are used along with deliberative means-ends planning methods to derive an initial suboptimal procedure for performing the task. In subsequent problem solving episodes, the declarative trace of past problem solutions are then used to develop procedures that are more optimal for rapidly reacting to the environment. We compare the predictions of our model with the behavior of Ackerman's (1988)[2].

The Kanfer-Ackerman ATC Task

Description of the Task

Although the Kanfer-Ackerman ATC Task is described in detail in other places (e.g. Ackerman, 1988; Ackerman & Kanfer, 1994), we briefly go over the task here. The Kanfer-Ackerman ATC task display consists of the following display elements (see Figure 1): (a) twelve hold pattern positions, (b) four runways, (c) feedback information on current score and penalty, conditions of the runways, wind direction and speed, (d) a queue stack with planes waiting to enter the hold pattern, and (e) two message windows (one for notification of weather changes, shown, and one for providing feedback on errors, not shown). The twelve hold pattern positions are divided into three levels corresponding to altitude, with hold level three being the highest and hold level one being the lowest.

Six rules govern this task: (1) Planes must land into the wind, (2) Planes can only land from hold level one, (3) Planes can only move one hold level at a time, but to any open position on that level, (4) Ground conditions and wind speed determine the runway length required by different plane types (5) Planes with less than 3 minutes of fuel left must be landed immediately, and (6) Only one plane at a time can occupy a runway. A weather change occurs approximately every 30 seconds, and planes enter into the queue approximately every 7 seconds.

The Kanfer-Ackerman ATC task is composed of three unit tasks: (1) accept planes from the queue into a hold pattern, (2) move planes within the three hold levels, and (3) land planes on a runway. The ↑ and ↓ keys move the cursor up and down between the different hold positions and runways, the **F1** key accepts the planes from the queue into a holding pattern, and the ↵ key selects a plane in the hold, places the selected plane into an empty hold position, or lands a plane on the runway. Cumulative score is calculated as follows: a) 50 points for landing a plane, b) minus 100 points for crashing a plane, c) minus 10 points for violating one of the six rules that govern the task.

[1]Kanfer-Ackerman Air Traffic Controller Task© program is copyrighted software by Ruth Kanfer, Phillip L. Ackerman, and Kim A. Pearson, University of Minnesota.

[2]This data set is indexed as study #2 in Ackerman's CD-ROM (Ackerman & Kanfer, 1994)

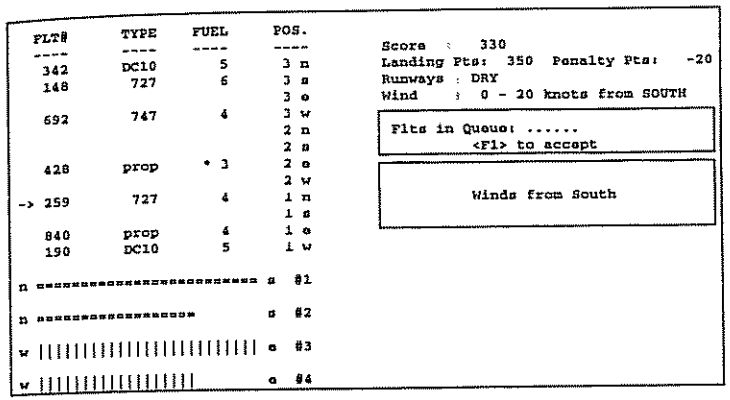

Figure 1. The Kanfer-Ackerman Air Traffic Controller© Task.

Task Instructions

Subjects are given sixty-one pages of computer-based instructions on the ATC task. The instructions consists of four sections: (1) how to move a plane between the hold levels, (2) how to land a plane, (3) how to accept a plane from the queue, and (4) the rules that govern the ATC task. The first three sections can be viewed as descriptions of the three unit tasks that underlie the ATC task. In the instructions for these three unit tasks, subjects are first presented with a single instruction page listing the subgoals required to complete that particular unit task. This page is followed by a series of pages describing a step-by-step example of performance on that unit task. For instance, the instruction page in Figure 2 is presented followed by an example of landing a plane. After introduction to each unit task, subjects are presented with the six rules that govern the ATC task (see Figure 3 for example).

ACT-R Representation of the ATC Task

Declarative Representation of the Task Instructions

We encoded the instructions for the three unit tasks as step-by-step declarative instructions in ACT-R. For example, instructions on how to land a plane is encoded as a set of six steps: (1) find a plane you want to land, (2) move to that plane, (3) select that plane, (4) find a runway to land, (5) move to that runway, and (6) select the runway. We encoded

```
To land a plane:

1) Press the ↑ or ↓ key until the arrow
   on the screen points to the plane you
   want to land.
2) Press the ↵ key to select the plane.
3) Press the ↑ or ↓ key until the arrow
   on the screen points to the desired
   runway.
4) Press the ↵ key again to land the
   plane.

        PRESS <SPACE BAR> TO CONTINUE
```

Figure 2: An instruction page on how to land a plane

```
RULE 1: PLANES MUST LAND INTO THE WIND.
        (That is, if the wind is from
        the South, the plane must be
        landed on a N-S runway.
        [DIRECTION]

RULE 2: PLANES CAN ONLY LAND FROM LEVEL
        1 [LEVEL]

RULE 3: PLANES IN THE HOLD PATTERN CAN
        ONLY MOVE 1 LEVEL AT A TIME.
        BUT TO ANY AVAILABLE POSITION
        IN THAT LEVEL [HOLD]

        PRESS <SPACE BAR> TO CONTINUE
```

Figure 3: An instruction page on task rules 1 - 3.

the knowledge of how to land a plane as six steps instead of four (as shown in Figure 2), because two steps are implicit in steps 1 and 3 in Figure 2. The first implicit step is finding a desired object, e.g. plane and runway, and the second implicit step is moving to that object. Furthermore, we encoded the rules of the task as declarative knowledge specifying constraint information.

Performing the Task Initially with Weak Methods

When our model first attempts to perform the ATC task, it does not have any procedures that it can use. This situation is analogous to that faced by a subject who must perform the ATC task for the first time, having just completed reading the instructions. The model begins by searching in its declarative memory for knowledge of how to perform the ATC task.

The model retrieves from memory a declarative encoding of the goal of the ATC task (i.e. to land planes). Then, the model searches for relevant declarative knowledge on landing planes. The model retrieves the encoding of the step-by-step instructions on how to land planes. The model uses the analogy mechanism (a built-in weak problem solving method in ACT-R) to proceduralize the declarative knowledge for landing planes. This creates a production rule[3] of the form given in Figure 4. Once this production rule is created, the model can now apply it to the current goal of landing a plane.

After the production rule fires, the model must satisfy several subgoals. First, the model must find a suitable plane to land. Once again, the model lacks a ready method that can satisfy this subgoal directly. The model uses a set of domain general means-ends planning production rules. An example of a domain general planning production rule is given in Figure 5. This production rule (and others associated with it) encapsulates domain general knowledge to which we assume people have access. The production rule basically states that if the goal is to select an object in a category and we know of an object that belongs to that category, then check if it is OK to choose that object. For example, if the goal is to select an object of the category "plane", and we know of an object of that category, e.g. flight # 347, then set a subgoal to check if flight #347 satisfies all known constraints on planes. If flight #347 fails to satisfy a constraint, the production rule will continue to try out different objects of the "plane" category until one is found that satisfies all the constraints.

```
IF the goal is to land a plane

THEN set a subgoal to find a suitable plane
     set a subgoal to move to the plane
     set a subgoal to select the plane
     set a subgoal to find a suitable runway
     set a subgoal to move to the runway
     set a subgoal to select the runway
```

Figure 4: Initial production rule for landing a plane.

[3]ACT-R uses production rules to represent procedural skills.

```
IF the goal is to select an object of a
     particular category
   and there is an object of that category
   and the object has not been tried before

THEN set a subgoal to check if the object
       satisfies all the constraints
```

Figure 5: A general production rule for selecting an object.

After finding a suitable plane, the model subgoals to move to that plane and select it. This process repeats to search for a suitable runway. Different runways are tried until one is found that satisfies all the constraints (such as rule 6, which states that the runway must be empty). After finding a suitable runway, the model subgoals to move to that runway and selects it.

Optimizing Procedures With Declarative Memory Trace of Past Problem Solving Episodes

A key assumption of our model is that subjects have access to the declarative trace of their problem solving episodes. If a subject just landed a plane, it seems reasonable that she can recall from memory the steps used to satisfy the subgoals for landing that plane.

Our model uses declarative trace of past problem solving episodes to build more efficient procedures. Specifically, optimization of production rules occurs through the process of replacing subgoals on the action side of the production rule with pattern matching on the condition side of the production rule. Figure 6(a) lists an example of an inefficient production rule, and figure 6(b) lists an example of an optimized production rule.

```
IF the goal is to land a plane

THEN set a subgoal to find a suitable plane
     set a subgoal to move to the plane
     set a subgoal to select the plane
     set a subgoal to find a suitable runway
     set a subgoal to move to the runway
     set a subgoal to select the runway
```

(a)

```
IF the goal is to land a plane
   and there is a plane in hold one
   and there is an open runway

THEN move to that plane
     select that plane
     move to that runway
     select that runway
```

(b)

Figure 6: (a) An example of an inefficient rule, and (b) An example of an optimized rule.

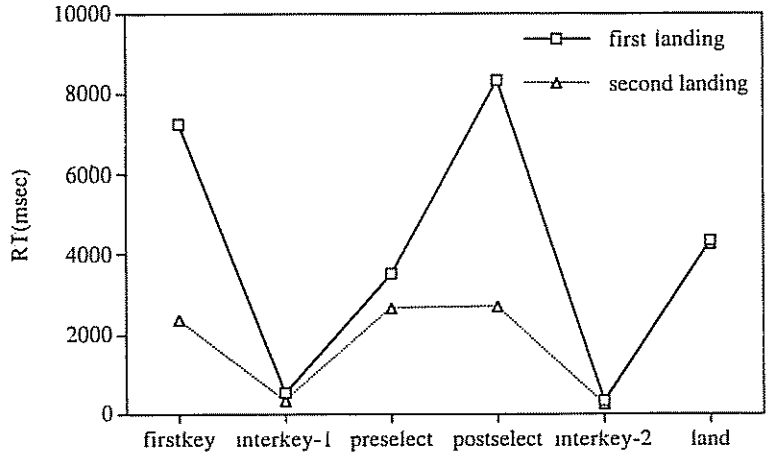

Figure 7: ACT-R Model's Latency Predictions for Landing a Plane.

Optimization is a process by which production rules become task-specific through experience. It is similar in spirit to Anderson's (1982) knowledge compilation process, but the scope of the our optimization process is much smaller. The logic behind the optimization is that in the process of landing the first plane, one must slowly reason out what constitutes a "suitable" plane and runway. However, after having reasoned this out the first time, one should be able to capitalize on this knowledge to narrow down the search space in choosing a plane and a runway in subsequent plane landings. In particular, one should only consider planes in hold one and runways that are open, along with other constraints.

To reiterate, the model initially performs the ATC task by using deliberative, domain general planning methods. Later, the model capitalizes on the knowledge gained through deliberative planning methods to develop more specialized, task-specific production rules. As we will show, this process of procedural skill learning that we are proposing makes some specific predictions about the timing of plane landing.

Predictions from the Model

When a trial begins, the arrow starts at the highest slot, "3n", in hold level three. The quickest way to land a plane from this point is to move to a plane in hold level one, select it, move to a runway, and then land. We can break up the time to land the first plane into a sequence of six events. First, the time between the beginning of the trial to the first down arrow pressed, *firstkey*. Second, the mean time to press a down arrow key when moving to a plane in hold level one, *interkey-1*. Third, the time between the last down arrow key pressed in moving to the plane and the enter key pressed to select the plane, *preselect*. Fourth, the time between the enter key pressed to select the plane to the first down arrow key pressed to move to the runway, *postselect*. Fifth, the mean time to press a down arrow key when moving to the runway, *interkey-2*. And finally, sixth, the time between the last down arrow key pressed in moving to the runway and the enter key pressed to land the plane, *land*.

The first time the model attempts to land a plane, it uses the deliberative production rule described in Figure 4. To land a second plane, the model uses the optimized production rule described in Figure 6. Figure 7 plots the predicted latency for the six events, for the deliberative production rule used for first landing and the optimized production rule used for second landing. As shown in the figure, the model predicts that the deliberative production rule will take longer for *firstkey*, where the deliberation for choosing a plane takes place, and *postselect*, where the deliberation for choosing a runway takes place. This is because the model must reason out what are "suitable" planes and runways for the first landing of a plane. However, for the second landing of a plane, the model uses the constraint knowledge gained from the first landing of the plane to construct a more task-specific production rule that has the constraint information built into the pattern matching side of the new rule and thereby eliminating the need to perform the deliberate problem solving.

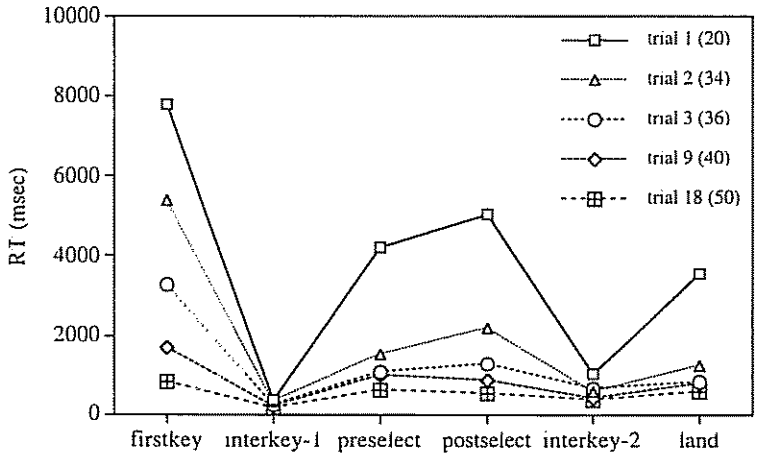

Figure 8: Latency data of subjects from Ackerman (1988).

In addition, shown in Figure 7, the model also predicts that acceleration in learning will be concentrated at the two points where long deliberation takes place during initial problem solving. This is because in subsequent landings, the model is able to build more task-specific production rules with constraint information on the pattern matching side of the production rule, thus eliminating the need to perform deliberative planning. In particular, while attempting to identify what are "suitable" plane and runway, the model realizes, for example, that all planes must be in hold one before one can land it. The model uses this knowledge to develop a new production rule in which this knowledge is built into the pattern-matching side of the production rule. Hence, the constraint knowledge gained through long deliberation during the first problem solving episode is used to build a production rule that automatically selects planes in hold level one.

Analysis of Subject Data

We used data from 58 of Ackerman's (1988) 64 subjects in our analysis. We excluded data from 5 subjects who did not complete at least eighteen (10-minute) trials, and also data from 2 subjects due to an error during their decompression from the Kanfer-Ackerman CD-ROM (Ackerman & Kanfer, 1994).

Figure 8 plots the mean latency for the six key events for those subjects who were able to successfully land the first plane at the beginning of each trial with only a two key deviations from the shortest key sequence necessary for the landing (e.g. up-arrow key). The number enclosed in parentheses next to the trial number shows the total number of subjects meeting this criterion.

As shown in Figure 8, the latency profile for the six events in trial 1 is similar to that predicted by the first landing latency profile in our model (Figure 7). The only difference is that our model predicts a larger contrast between latency time for *preselect* and *postselect*, whereas in the subject data, these two events show a similar latency profile. A reasonable explanation for this is that within our model, the entire burden of deliberation for choosing a runway is placed in the *postselect* event. In reality, however, it is more likely that some subjects are choosing a runway before they press the "enter" key. Since the data plotted for trial 1 in Figure 8 is aggregated over 20 subjects, this may reflect an aggregation of two types of subjects: (1) subjects who look for a runway before they press the "enter" key, and (2) subjects who take the time to look for a runway after they press the "enter" key. Hence, if we average over *preselect* and *postselect* latency time, the model predicts a little less than six seconds of latency, which is consistent with the subject data.

In addition, the prediction of our model that acceleration in learning will be concentrated at the points of deliberation is also borne in the subject data shown in Figure 8. As can be seen, the acceleration in learning is occurring precisely at the *firstkey* event, where the deliberation for finding a plane takes place, and at the *preselect* and *postselect* events, where the deliberation for finding a runway takes place.

However, while it is clear that the first landing of the model is comparable to the first landing of subjects in trial 1, the model is less capable of predicting second landing. In particular we forced the model to use the optimized production to land the second plane, in order to examine the implications of the procedure optimization process that we have posited. As for the subject data in Figure 8, which indicates a gradual learning process, we hypothesize at least two types of processes at work. First, while the production may be learned, it may not necessarily fire the next time. Instead, it may take time for the procedure to gain enough strength to fire. The second, the aggregation over subject data may hide the differential rate at which subjects are learning and using the optimal production rule. Individual differences in the rate at which subjects acquire and use the efficient production rule likely vary among different people. The purpose of our modeling effort was to describe a procedural optimization process which can account for the qualitative acceleration in learning in dynamic tasks like the ATC task.

Conclusion

The ideas elaborated in our model are consistent with past research in acquisition of procedural skill from instructions. Catrambone (1990, 1995) has examined how procedural skills are acquired from instructional text. In particular, he differentiates between specific and general knowledge. He argued that general knowledge is more important, because it transfers better to novel tasks. However, specific knowledge is faster than general knowledge in its application (Catrambone, 1990). Since speed is critical in performing dynamic, time-constrained tasks (Agre & Chapman, 1987), it is precisely the acquisition of such task-specific procedural skills that becomes critical in dynamic tasks.

In addition, our use of a memory trace for past problem solutions is also consistent with other research efforts like Hammond's (1990) case-based planning. Hammond (1990) has argued that people retrieve past plans and modify it to solve the current problem. However, our process differs from case-based planning, because we use the declarative trace of past problem solutions to develop more efficient procedures. Learning in case-based planning is fairly limited, for example, to accumulation of plans in memory.

In the ATC task and other dynamic tasks, people become highly reactive just as the situated theorists have argued. However, we have shown that people actually go through a more deliberative stage in which they spend considerable time reasoning about constraints. Indeed, these two situations are not mutually exclusive. They may instead define two endpoints of a learning continuum in dynamic tasks.

Our purpose, therefore, has been to try to provide a bridge between deliberative problem solving methods (e.g. Newell & Simon, 1972) that are necessary in initially performing in a novel task domain, and the need to react quickly to the environment, as posited by theory of situated activity (e.g. Agre & Chapman, 1987). ACT-R is a model that is capable of producing this transition from deliberation to reaction (see John & Bauer, 1995 for an alternative model within the Soar architecture for skill learning in dynamic task). It does so by learning from declarative traces of its initial problem solving efforts. Moreover, it is capable of making predictions about landing times and their speed-up which are approximately in the ball park of the observed times.

Acknowledgments

The research reported in this paper was supported by the Office of Naval Research, Cognitive Science Program, under Contract Number N00014-95-1-0223 to Lynne M. Reder and John R. Anderson. We would like to thank Kevin Gluck and Doug Thompson for comments on earlier drafts of this paper.

All correspondences should be addressed to Frank J. Lee at the Department of Psychology, Carnegie Mellon University, Pittsburgh, PA 15213-3890.

References

Ackerman, P. L. (1988). Determinants of individual differences during skill acquisition: Cognitive abilities and information processing. *Journal of Experimental Psychology: General, 117*, 288-318.

Ackerman, P. L., & Kanfer, R. (1994). Kanfer-Ackerman Air Traffic Controller Task© CD-ROM Database, Data Collection Program, and Playback Program. Office of Naval Research, Cognitive Science Program.

Agre, P. E., & Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 268 - 272.

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review, 89*, 369-406.

Anderson, J. R. (1993). *Rules of the Mind*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Catrambone, R. (1990). Specific versus general procedures in instructions. *Human Computer Interaction, 5*. 49-93.

Catrambone, R. (1995). Following instructions: Effects of principles and examples. *Journal of Experimental Psychology: Applied, 1*, 227-244.

Hammond, K. (1980). *Case-Based Planning: Viewing Planning as a memory Task*. New York, NY: Academic Press.

John, B. E. & Bauer, M. I. (1995). Modeling Time-Constrained Learning in a Highly Interactive Task. In *Conference on Human Factors in Computing Systems (CHI'95)*, 19-26.

Newell, A., & Simon, H. A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice Hall.