Wang, S., & Ritter, F. E. (2023). Modeling multiple strategies and learning in a complex fault-finding task. *16th International Conference on Social Computing, Behavioral-Cultural Modeling, & Prediction and Behavior Representation in Modeling and Simulation (SBP-BriMS) 2023*. 23-BRIMS-086.

Modeling Multiple Strategies and Learning in a Complex Fault-Finding Task

Shan N. Wang and Frank E. Ritter

Penn State University, State College PA 16801, USA

Abstract. Previous work has modeled strategies in a simple fault-finding task. We present multiple models of strategies that people apply to find faults in a complex circuit, with and without learning. We continued modeling multiple strategies for tasks with higher complexity. The multiple strategies are implemented in Python with a novel approach that uses hierarchical task analysis, the Keystroke-Level Model (KLM), and the power law, to predict performance time. To evaluate those models, we used human data from a large study (Ritter et. al, 2022). We model the test session data when participants had more time to learn and develop their strategies. We developed four strategies by analyzing the top 6 participants who had 100% correct rate in the test session. We then compared the human performance time and the prediction time by our strategy models, with and without learning. The strategies can predict 62% of the participants. We provide insights into why we sometimes failed to predict performances well, such as not modeling errors nor strategy switches.

Keywords: Individual differences, Cognitive modeling, Learning.

1 Introduction

It is possible to construct a learning model that can capture individual differences in strategies as these strategies develop. We can find predictable sources of variability in learning across time and across learners. Modeling human behavior using different strategies and comparing them with the time course of individual learning behavior remains difficult and rare, but not impossible. A model that learns helps describe how humans use knowledge to solve problems and how the repetitive application of the same knowledge leads to faster performance. A good overview of the categories of human learning is provided by [2]. These models have had their predictions compared to averaged data but have not had their predictions compared to individual learners. A problem in the study of complex problem solving, especially in a learning context, is the spectrum of individual differences. Understanding individual differences can help understand the acquisition and application of complex skills. Averaging over strategies will distort results and hide important and reliable effects [7,19]. Models of individual differences have been built by previous researchers in various ways, such as differences in global system parameters [4,5,9], differences in knowledge [1,3,10], and in how differences arise through learning[12,18,19].

Thus, previous work has argued for the need and the usefulness of representing strategies [15,17,20]. These models have shown that individual differences in problem solving arise in multiple tasks, and that modeling individual strategies can lead to a better understanding of how the task is performed and how learning strategies arise [6,21]. Friedrich and Ritter [8] also presented the case that strategies are important for understanding behavior, and that learning is important as a component to shift between and to evolve strategies. They pointed out that little research had been done on modeling different strategies for a task, because to do so, tasks need to be complex enough to allow multiple strategies, and individual behavior must be traced and analyzed, and multiple strategies must be modelled. To extend this line of work, we report how we modeled multiple strategies while they were being learned, how they matched participants' performance (some fairly well), and how they inform us about behavior, transfer, and learning.

We have presented a small start of this work, one strategy, as a poster in ICCM [22], this paper has four strategies with more details and analysis.

1.1 The MENDS Task

We used a complex electrical troubleshooting task to study problem solving, with a much more complex system, the BenFranklin Radar System that has five times more components. Figure 1 shows the BenFranklin Radar system schematic. It has 36 components, including a power supply that does not break: five times as many components as the Klingon Laser Bank task.

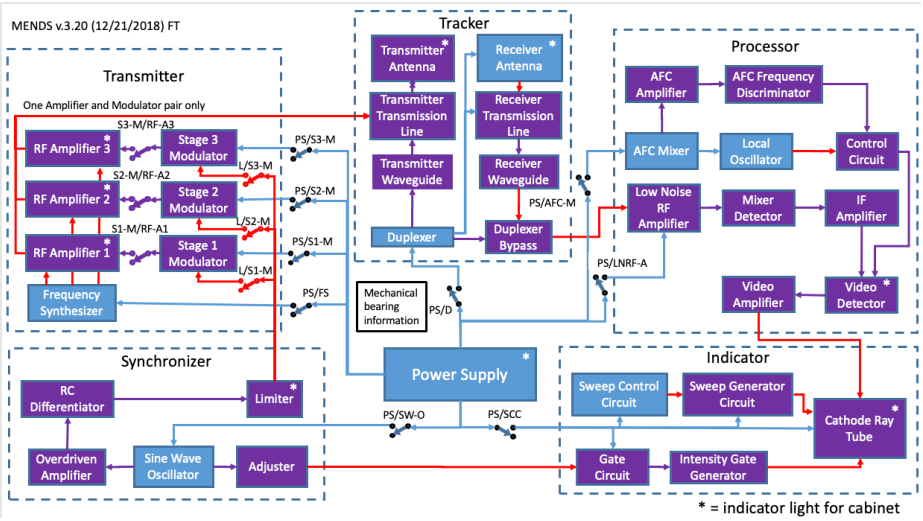

Fig. 1. The Schematic for the BenFranklin Radar. Blue lines are power; red lines are signal; purple lines are both.

MENDS, in Figure 2, is a simulator created by Charles River Analytics for the BenFranklin Radar system, used under license. In MENDS, participants can click and open

the subsystem (trays) to see the components in each subsystem. Based on their schematic knowledge, the light and switch conditions, participants can decide and click the component that they think is the broken component. Components without power are in grey.

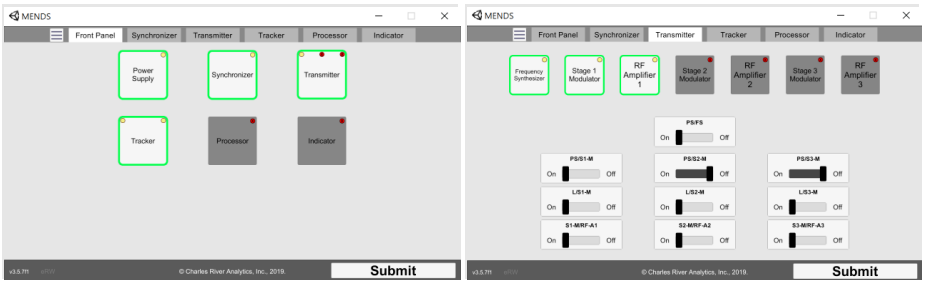

Fig. 2. The MENDS simulator's front panel and one subsystem.

1.2 Human Performance Data

We collected participants' mouse moves for our modeling and data analysis using the RUI logger [11,13]. To gather data, a user study was run [14]. The goal (in addition to studying learning and retention on a complex task) was to identify strategies with a larger number of participants. In the larger study, sessions 1, 2 and 4 are practice sessions; Session 5 is the test session. We used data from the test session when participants tend to have developed their strategies. After data cleaning, we had 111 participants' data in the test session. Participants have 5 minutes in Session 1 to practice MENDS and are asked to finish 20 problems in the test session. The number of finished tasks varied, because not all participants finished 20 problems in the test session.

We collected the component and times the participants clicked on. From the mouse clicks of the top 6 participants in the test session, we developed and implemented four strategies in the BenFranklin Radar task. The observed time for participants' performance was compared with the predicted time of our strategy models, without and with learning.

In the remainder of the paper, we present how we modeled the MENDS task, followed with the descriptions of the current four strategies (the grey upstream strategy as an example of the technical details) and comparison among strategies. Further, we present a comparison of human performance and model predictions without and with learning. Finally, we present an example participant's match to show how well our strategy model predicts their time. (Participant IDs ran from 100-500 and represented different running sessions).

2 Modeling the Task and Strategies

We present a flowchart, Figure 3, to show how we built a simple task model for MENDS in Python [16]. The simple task model used Panda data frame to store and reflect components' broken status (1: component is fine; 0: broken), light status (1:

component has light on; 0: light off), downstream components to give power, upstream components to receive power, switch condition before them (1: switch on; 2: switch off), the number of times the required schematic knowledge have applied by participants.

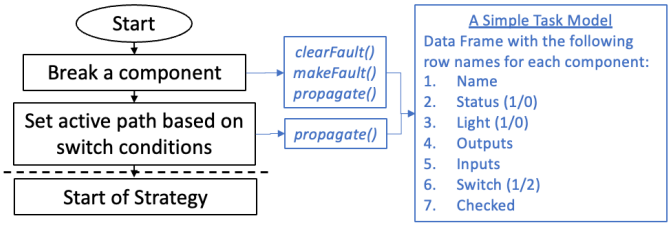

Fig. 3. A flowchart for the simple task model and the grey-upstream strategy model. An active path refers to the components receiving power and that are supposed to have lights on.

By analyzing mouse moves of participants who had 100% correct rate in the test session, we came up with four strategies that may be used to identify the broken component. Here we define the strategy in the scope of a task. Those strategies are categorized by four features: their starting point, how the front panel information was used, degree of schematic knowledge used, and degree of interface information used. Variations within one strategy are also possible. All strategies find the correct fault.

2.1 The Grey Upstream Strategy

The grey upstream strategy (GreyUp), shown in Figure 4, is based on participants PID 324, 420, 451, 453. The strategy involves two major steps, finding a grey component as a starting point and tracing upstream in the schematic till finding the broken component. To locate the starting point, users click into the first grey tray using light information from the front panel and identify the first grey component by interface order from left to right and up to down within the clicked tray. Starting from the first grey component, participants use their schematic knowledge to trace up until they identify the broken component, which is the only one with its light off but all its upstream components in the active path are with their lights on.

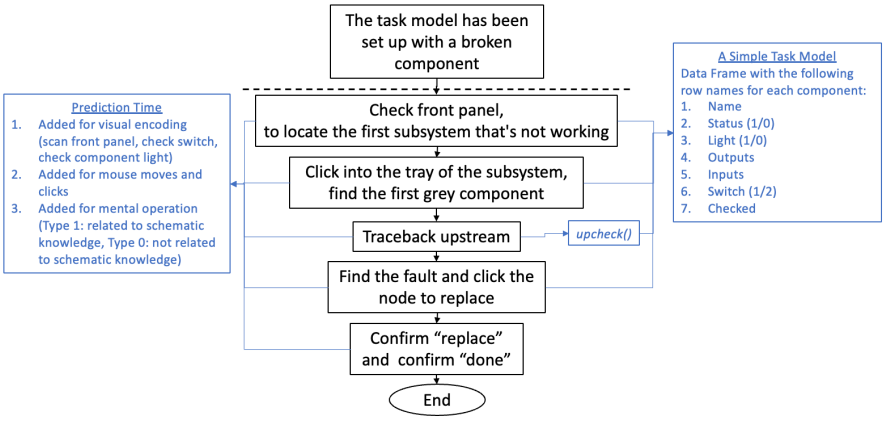

Fig. 4. Flowchart of the Grey Upstream strategy.

2.2 The One-by-One First Grey Strategy

The one-by-one first grey strategy (OByO) was developed based on PID 448. The strategy walks through trays one by one and stops and clicks the component when the first grey component in the active path is found. This strategy ignores front panel information and uses only interface information.

2.3 The All-Trays Strategy

The all-trays strategy (AllTrays) was developed based on PID 347. The strategy walks across all trays while deciding the fault, then directly goes to a certain tray to locate the broken component. Components were assigned a number based on their distance from the power supply. The smaller the number, the closer the component to the power supply. The one that has the smallest number among the grey components is the broken one. This strategy ignores front panel information and uses both schematic knowledge and interface information.

2.4 The Smaller Lights Strategy

The smaller lights strategy (SLights) was developed based on PID 396. The strategy uses smaller indicator lights on the front panel to decide the first tray to go; components are grouped by smaller indicator lights. By the combination of indicator lights on the front panel, participants know which tray the broken component is in, then directly go to that tray to click it. This strategy uses smaller indicator lights on front panel and uses both schematic knowledge and interface information. This strategy requires better and higher-level use of schematic knowledge.

2.5 How the Models Predict Time and Learn

The models take steps to solve the task based on each strategy, and time is added with each type of action. The time for each step depends on the learning level. The times added with no learning condition are shown in Table 1. Under the learning condition, we assume that knowledge is transferred to later tasks and the participants solve the

later tasks faster based on the repetitive application of the same knowledge. Time parameters used in the models include visual encoding, 0.4 (s); mouse move, 1.1; mouse click, 0.2; mental operation, 1.35; learning rate, 0.4.

Models learn by modifying the mental operation time. We do this by having mental operation time as a function and not a constant. The function *mental(type, var2)* gives different times for different types of mental operation and learning levels. The *type* variable includes *type 1* and *type 0*. *Type 1* refers to the mental operation time retrieving a component by using schematic knowledge. *Type 0* refers to any other mental processing not related to schematic knowledge. *Type 1*, retrieving a component, can be faster with learnings which indicated by *var2*. For *type 0*, *var2* is a random number because the time is assumed to be fixed, and no learning happens. For situations that assume no learning happens, the mental operation function in models uses constant 1.35 s. For situations that assume learning happens, the mental operation function in models uses the following formula:

$$Prediction\ Time\ = 1.35 \cdot\ n^{-l} \tag{1}$$

Where n is the number of times the component has been checked or the number of times the required knowledge of circuit has being used. The value of *l* represents the learning rate, 0.4.

2.6 Comparison among the Four Strategies

Those four strategies have different starting points and use different degrees of schematic knowledge and interface information. They take different times to finish the same task. Figure 5 shows the models' predicted times on the 35 faults as if they solved each for the first time (without learning). The figure shows that the strategies are different, and that their task times also vary by different tasks.

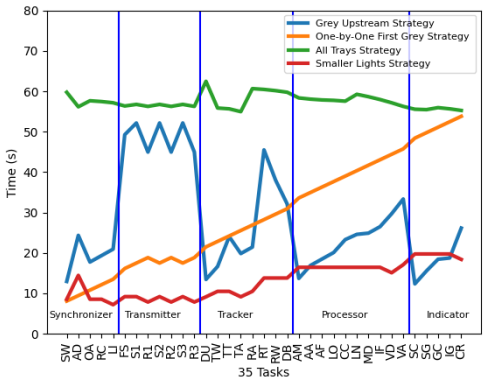

Fig. 5. Strategy predictions for the 35 faults without learning across problems. The x-axis lists all the 35 faults (not in completion order of any session).

Figure 6 shows the predictions of the strategies for the 20 problems without learning in the test session, showing again that the strategies are different: they predict different time for the same task; one strategy also predicts different times on different tasks.

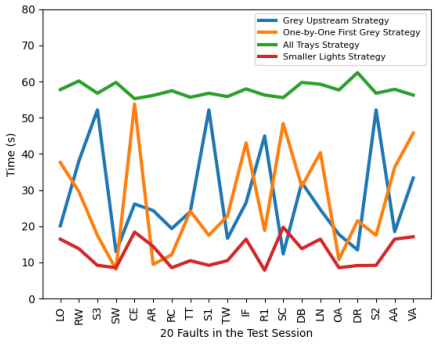

Fig. 6. Strategy predictions for the 20 problems without learning in the test session. The x-axis shows the same order of the 20 tasks in the test session.

3 Comparing Human Performance and Model Predictions

We next discuss the results of the comparison between the strategies and participants, without and with learning. The developed strategies were compared to the human performance data in the test session, with no learning and learning. The models were used to solve tasks in the test session. We trained the models with the previous sessions that participants experienced before we run the models for tasks in the test session under learning. The predicted times are then compared to the observed times of the participants. We consider that participants fit well to a strategy if they have R^2 with a p-value $< .05$.

3.1 Without Learning VS With Learning

14 participants' behaviors match a strategy ($p < .05$; R^2 varied) without learning in the test session. 69 participants' behaviors match a strategy with learning in the test session. Table 2 shows only part of the data, under the no learning condition, as an example. Table 3 shows only part of the data, under the learning condition, as an example. We caught some the strategies participants used, such as the grey upstream strategy (GreyUp) and all trays strategy (AllTrays). The one-by-one strategy (OByO) and the smaller light strategy (SLight) did not match well. Negative correlations are shown as 0's in the table for clarity.

Table 2. Without learning. Example well-fit participants, the four strategies, and their corresponding R^2. Those with p-value $< .05$ have a * attached.

PID	AllTrays	GreyUp	OByO	SLight
378	.322*	0	.024	.026
387	.067	.315*	0	0
404	.021	.233*	0	0
413	0	.492*	0	0
429	0	.342*	0	0

Table 3. With learning. Example well-fit participants, the four strategies, and their corresponding R^2. Those with p-value < .05 have a * attached.

PID	AllTrays	GreyUp	OByO	SLight
361	.763*	.082	0	0
407	.737*	.012	.033	.019
350	.594*	.095	.055	.012
355	.577*	.062	0	0
352	.572*	.078	.004	.017

3.2 An Example: PID 413

Figure 7 shows the match of PID 413 as an example. PID 413 has R^2 of .498, without learning and an R^2 of .518, with learning for the gray upstream strategy. The red dotted line is the observed time from human data; the blue solid line is the predicted time without learning; the black solid line is predicted time with learning and was trained with previous sessions' faults.

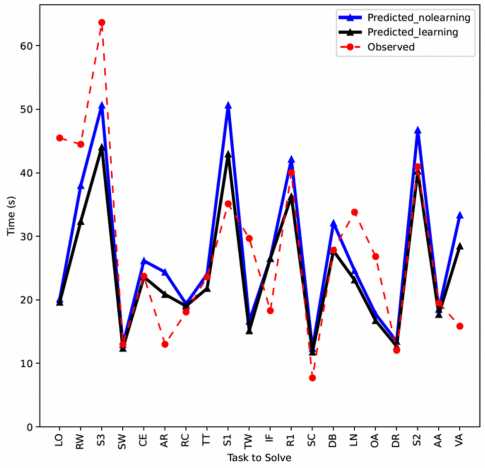

Fig. 7. An example of PID 413. A comparison of human data, our Grey Up-stream Strategy model with learning, and the Grey Up-stream Strategy without learning.

4 Discussion and Conclusion

We modeled four strategies in problem solving and in different learning conditions. We hope to urge an awareness of understanding diversity of behavior. Different strategies

have different starting points, different ways of using schematic knowledge, and interface information. They took different times, and all found the correct solution. With the assumption of only one standard strategy, we may mistakenly regard many of the participants as being misbehaved when they apply a different strategy. In a worse case, we may interrupt them at the wrong time, believing that we are correcting and helping them.

One should indeed be careful averaging behaviors. Participants use different strategies within this task. Averaging over strategies will distort results and hide important information. We identified and used four strategies for the complex fault-finding task to predict human performance. More than four strategies exist, because some negative correlations are also significant. Further exploration may come up with some counter-intuitive strategies. We also have built a fifth strategy called random-start strategy. This strategy needs further development to run all the possible starting points to calculate correlations of prediction time and human performance time.

The current strategy models we have are from the top 6 well-performed participants. The rest of the participants made much more errors during the fault-finding process. If our strategy models consider errors, the match of participants and strategies may increase. We have built a simple error model and indeed got higher R^2 [23]. More and complicated error models are worth further explored. Moreover, currently, we assume that participants were using the same strategy across a session, but some participants may have applied various strategies within one session for different tasks. We also assume that they apply the strategy perfectly without variation. It requires them to have necessary working memory and schematic knowledge to help them finish all the steps of a strategy. In this case, we have not modeled errors, or lapses, or changes of strategies within the same session. Modeling those can be our future steps.

References

1. Yuichiro Anzai and Herbert A. Simon: The theory of learning by doing. *Psychological Review* 86, 2: 124–140 (1979).
2. F. Gregory Ashby and W. Todd Maddox: Human category learning. *Annual Review of Psychology* 56, February: 149–178 (2005).
3. B.J. Best and Marsha Lovett: Inducing a cognitive model from examples provided by an optimal algorithm. In *Proceedings of the Seventh International Conference on Cognitive Modeling*, 56–61 (2006).
4. Stuart K. Card, Thomas P. Moran, and Allen Newell: *The Psychology of Human-Computer Interaction*. CRC Press (1983).
5. Larry Z. Daily, Marsha C. Lovett, and Lynne M. Reder: Modeling individual differences in working memory performance: A source activation account. *Cognitive Science* 25, 3: 315–353 (2001).
6. Peter F. Delaney, Lynne M. Reder, James J. Staszewski, and Frank E. Ritter: The strategy-specific nature of improvement: The power law applies by strategy within task. *Psychological Science* 9, 1: 1–7 (1998).
7. Renée Elio and Peternela B. Scharf: Modeling novice-to-expert shifts in problem-

solving strategy and knowledge organization. *Cognitive Science* 14, 4: 579–639 (1990).
8. Maik B. Friedrich and Frank E. Ritter:. Understanding strategy differences in a fault-finding task. *Cognitive Systems Research* 59: 133–150 (2020).
9. Sue E. Kase, Frank E. Ritter, Jeanette M. Bennett, Laura Cousino Klein, and Michael Schoelles: Fitting a model to behavior reveals what changes cognitively when under stress and with caffeine. *Biologically Inspired Cognitive Architectures* 22: 1–9 (2017).
10. David E Kieras and David E Meyer: The role of cognitive task analysis in the application of predictive models of human performance. In *Cognitive Task Analysis*. 251–274 (2020).
11. Urmila Kukreja, William E. Stevenson, and Frank E. Ritter: RUI: Recording user input from interfaces under Windows and Mac OS X. *Behavior Research Methods* 38, 4: 656–659 (2006).
12. Jill H. Larkin, John McDermott, Dorothea P. Simon, and Herbert A. Simon:. Models of competence in solving physics problems. *Cognitive Science* 4, 4: 317–345 (1980).
13. Jonathan H Morgan, Chen-Yang Cheng, Christopher Pike, and Frank E Ritter: A design, tests and considerations for improving keystroke and mouse loggers. *Advance Access publication on* 6 (2013).
14. Frank E. Ritter, Farnaz Tehranchi, Mat Brener, and Shan Wang: Testing a complex training task. *Proceedings of ICCM 2019 - 17th International Conference on Cognitive Modeling*, January 2019: 184–185 (2020).
15. Frank E Ritter and Peter A Bibby: Modeling how, when, and what is learned in a simple fault-finding task. *Cognitive Science* 32, 5: 862–892 (2008).
16. Frank E Ritter, Deja Workman, and Shan Wang: Predicting learning in a troubleshooting task using a cognitive architecture-based task analysis. In *International Conference on Cognitive Modeling (2022)*.
17. Robert S. Siegler: The perils of averaging data over strategies: An example from children's addition. *Journal of Experimental Psychology: General* 116, 3: 250–264 (1987).
18. Robert S. Siegler: Strategy choice procedures and the development of multiplication skill. *Journal of Experimental Psychology: General* 117, 3: 258–275 (1988).
19. Robert S. Siegler: Individual differences in strategy choices: Good students, not-so-good students, and perfectionists. *Child Development* 59, 4: 833 (1988).
20. Herbert A. Simon and Stephen K. Reed: Modeling strategy shifts in a problem-solving task. *Cognitive Psychology* 8, 1: 86–97 (1976).
21. Kurt VanLehn.: Rule acquisition events in the discovery of problem-solving strategies. *Cognitive Science* 15, 1: 1–47 (1991).
22. Shan N. Wang and Frank E. Ritter: Modeling a strategy with learning in a complex task. In *International Conference on Cognitive Modeling (2023)*.
23. Shan N. Wang and Frank E. Ritter: Exploring errors towards a more realistic strategy model. In *International Conference on Cognitive Modeling (2023)*.