UC Merced
Proceedings of the Annual Meeting of the Cognitive Science Society

Title
Rule Learning and the Power Law: A Computational Model and Empirical Results

Permalink
https://escholarship.org/uc/item/2r37330b

Journal
Proceedings of the Annual Meeting of the Cognitive Science Society, 15(0)

Authors
Nerb, Josef
Krems, Josef F.
Ritter, Frank E .

Publication Date
1993

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

# Rule learning and the power law: A computational model and empirical results

**Josef Nerb** **Josef F. Krems**
Dept. of Psychology, University of Regensburg
D-8400 Regensburg, Germany
nerb@rpss3.psychologie.uni-regensburg.de
krems@rpss3.psychologie.uni-regensburg.de

**Frank E. Ritter***
Dept. of Psychology, University of Nottingham
Nottingham, NG7 2RD, England
ritter@psyc.nott.ac.uk

## Abstract

Using a process model of skill acquisition allowed us to examine the microstructure of subjects' performance of a scheduling task. The model, implemented in the Soar-architecture, fits many qualitative (e.g., learning rate) and quantitative (e.g., solution time) effects found in previously collected data. The model's predictions were tested with data from a new study where the identical task was given to the model and to 14 subjects. Again a general fit of the model was found with the restrictions that the task is easier for the model than for subjects and its performance improves more quickly. The episodic memory chunks it learns while scheduling tasks show how acquisition of general rules can be performed without resort to explicit declarative rule generation. The model also provides an explanation of the noise typically found when fitting a set of data to a power law — it is the result of chunking over actual knowledge rather than "average" knowledge. Only when the data are averaged (over subjects here) does the smooth power law appear.

## Introduction

From a psychological point of view *planning* can be considered a problem-solving activity in which, in a *prospective* manner, an ordered sequence of executable actions has to be constructed. In a more formal sense, this means specifying a sequence of operators with well-defined conditions and consequences that transform a given state into a goal state. For interesting problems, the entire problem space cannot be searched, and heuristics must be used to guide the search.

In scheduling problems, a specific, important subset of problem solving, the task of the problem solver is to find an optimal schedule based on the given constraints (e.g., minimal processing time). Factory scheduling (so-called job-shop scheduling) is a further subset of scheduling tasks, namely, to find the optimal ordering of activities on machines in a factory.

Job-shop scheduling has direct, practical importance. Over the last two decades algorithms have been derived that produce optimal (or near optimal) solutions for scheduling tasks using operations research techniques (e.g., Graves, 1981) as well as AI techniques (e.g., Fox, Sadeh & Baykan, 1989). One of the most popular systems, based on constraint-propagation, is ISIS (Smith, Ow, Potvin, Muscettola & Matthys, 1990). Other AI-based approaches have used the general learning mechanisms in PRODIGY (Minton, 1990) or Soar (Prietula, Hsu, Steier & Newell, 1992). These systems rely on the assumption that general methods for efficient problem solving can be discovered by applying a domain-independent learning mechanism.

In psychology, on the other hand, little is known about how scheduling is performed as a

*The third author was partially supported by a grant from the Joint Council Initiative in HCI and Cognitive Science, grant number SPG 9018736.

problem solving activity and about the acquisition of scheduling skills (for a counterexample and earlier call to arms, see Sanderson, 1989). One way to investigate how people acquire skills and knowledge in planning tasks is to use a general architecture for cognition like Soar (Newell, 1990) to construct and evaluate computational models of scheduling and the learning processes it incorporates. This approach provides methods for conceptualizing the problem solving situation and for implementing considerations on the knowledge level such as learning.

Thus the main goal of a cognitive approach (and this paper) is not to find efficient methods for scheduling, but to find models of skill acquisition that can claim cognitive plausibility. Here, we first find and describe several empirical constraints on scheduling, and then describe a computational model that was developed based on these constraints. Use of this model to interpret further data allows us to examine performance, including learning in this domain in a principled, detailed way. The result we include here is a knowledge-based explanation of noise in the power law of learning.

## Skill acquisition in scheduling tasks

### The Task

The task — for the subjects as well as for the computational model — was to schedule five actions optimally. The subjects had to act as a scheduler or dispatcher of a small factory, who's task it was to schedule jobs on two machines (A and B). Each job had to be run in a fixed order on both machines (first A then B) requiring different resources (processing time). The essential constraint for the subjects was to find that order of jobs that produced the minimal total processing time. Sets of five jobs with randomly created processing times were given to the subjects on a computer display. The subjects had to determine which out of five jobs should be run first, which second, and so on. For this kind of task an algorithm to find the optimal solution is available (Johnson, 1954). The general principle requires comparing the processing times of the jobs and finding the job using the shortest time on one of the two machines. If this is on machine A than the job has to be run first, if it is on B, then last. In order to get an optimal solution this principle has to be applied until all of the jobs are scheduled. Suboptimal sequences result if only parts of the general principle are used, e.g., only the demands of resources on machine A are used for ordering the jobs. This special task of modest complexity was selected because (a) it is simple enough to assess the value of each trial's solution by comparing it to the actual optimal solution, (b) the task is hard enough to be a genuine problem for subjects, who have to solve it deliberately, and (c) solving the task without errors requires discovering and applying a general principle.

### What is learned in this task?

Learning to solve scheduling tasks, like learning in general, requires the acquisition as well as the storage of rules in memory. In this task, acquisition is discovering the general rule or inferring at least useful scheduling heuristic rules while performing the task. If no rule on how to schedule jobs is available and the problem solver progresses through blind search, on average no great improvement should occur. Only if the subject generates internal hypotheses about the schedule ordering rules, and if feedback about the correctness of these assumptions is available, will the subject be able to discover efficient scheduling rules. And then, only if a discovered rule is stored in memory will the improvement be applied in later trials. As in *impasse-driven learning*-theories (VanLehn, 1988), it is assumed that rule acquisition particularly takes place when subjects face a situation in which their background knowledge is not sufficient to solve the problem immediately.

Of course, as in other domains, learning in scheduling tasks depends on the amount of practice and it is highly situated. An essential 'situational' factor, which facilitates or inhibits the acquisition of rules and thus the progress in learning, is the interaction of the problem solver with the environment. This gives feedback about the quality of the subject's problem solving solution and therefore about the efficiency of the applied rule.

In a recent study (Krems & Nerb, 1992) the influence of different types of feedback about the quality of solutions on the learning process was investigated. Subjects were given either (a) Quantitative information: how good a single solution is compared with the optimum or previous solutions of the problem solver. No information about the underlying rule or how the optimal solution can be found was given. Or (b) Qualitative information: an assessment of a solution in relation to the

optimal scheduling rule, which out of the $n$ jobs were correctly scheduled.

Subjects who received quantitative information about the distance of their own solution to an optimum more often discovered a good scheduling algorithm than those who received qualitative hints about which jobs were correctly scheduled. The qualitative hint subjects were much more oriented towards optimizing a single solution rather than getting insight on a more abstract level. Their behavior can be well described within repair-theory (VanLehn, 1988). The results point to distinct types of learning based on different kinds of feedback.

## Sched-Soar

Sched-Soar is a computational model of skill-acquisition in scheduling tasks. The architectural component is strictly separated from the task-specific knowledge. Soar, a candidate unified theory of cognition (Newell, 1990), was used as the general cognitive architecture because learning, particularly impasse-driven learning, is an essential part of this architecture (Rosenbloom, Newell, Laird & McCarl, 1991). Sched-Soar uses 401 productions implementing eight problem-spaces.

### Empirical constraints

The empirical constraints are taken from experiments (Krems & Nerb, 1992) where 38 subjects each created 100 schedules. Although the main focus of this study was to investigate the effect of different kinds of feedback on learning, the data also describe how long it takes to solve the task, what kind of learning occurs, and so on. The main empirical results used to constraint the design of our process model of scheduling skill acquisition are:

(1) Total processing time: The task takes 22.3 s, on average, for a novice (min-value: 16 s, max-value: 26 s.).
(2) General speed-up effect: On average, the processing time for scheduling a set of jobs decreased 22% from the first ten trials to the last ten.
(3) Improvement of solutions: The difference between the subject's solutions and the optimum decreased more than 50% over the 100 trials.
(4) Suboptimal behavior: It should be emphasized that even after 100 trials the solutions are not perfect.
(5) Algorithm not learned: None of the subjects, however, detected the optimal scheduling rules (i.e., nobody could give a verbal description of the underlying principle when asked after the trials).

### Model assumptions

In addition to the empirical constraints we include the following general constraints:

(1) The task is described and represented in terms of problem spaces, goals and operators, as a Problem Space Computational Model (Newell, 1990). All knowledge is implemented as a set of productions and chunking is used as a universal learning mechanism. Soar's bottom-up chunking mechanism was used for learning, which means that chunks were built only over terminal subgoals. This is proposed as characteristic of human learning (Newell, 1990, p.317).
(2) A minimum amount of knowledge about scheduling-tasks is initially provided as part of long-term knowledge (e.g., to optimize a sequence of actions it is first necessary to analyze the resource demands of every action). Also basic algebraic knowledge is included, such as ordinal relations between numbers.
(3) Feedback-driven: If the available internal knowledge is not sufficient to choose the next action to schedule, the supervisor is asked for advice. These are situations in which a human problem-solver would have to do the same, or to guess.

### Processing steps

Sched-Soar begins with an initial state containing the five jobs to be scheduled and a goal-state to have them scheduled well, but without knowledge of the actual minimum total processing time. The minimal scheduling knowledge that Sched-Soar starts with leads to these main processing steps, which are applied for every single scheduling step:

(1) Sched-Soar analyses the situation and tries to find a job to attempt to schedule.
(2a) If no decision can be made, despite examination of all available internal knowledge, Sched-Soar requests advice from the environment. The advice specifies the job that is the optimal choice to schedule next in the current set.
(2b) After getting advice about which job to schedule next, Sched-Soar reflects on why it applies to the current situation. In doing so, Sched-Soar uses

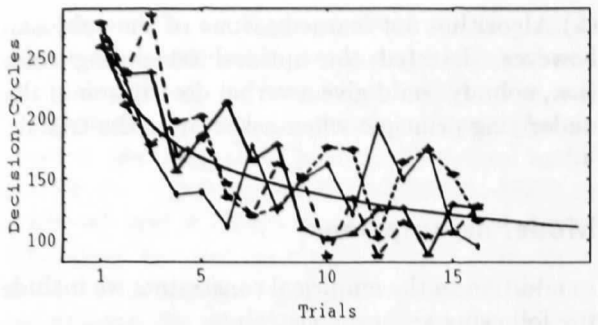


Figure 1: Individual data: Processing time in Soar decision-cycles (DCs) for four sets of simulated data for trials 1 to 16 and a power law fit.

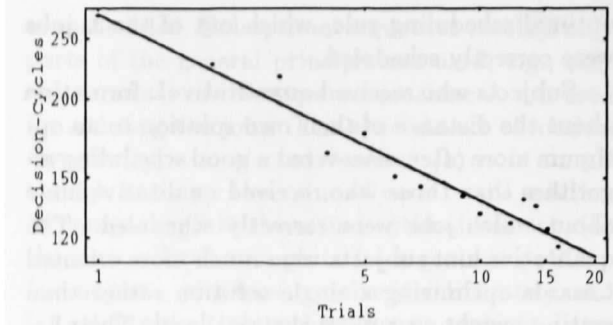


Figure 2: Averaged data: Processing time in DCs for four sets of simulated data for trials 1 to 16 and a power law fit.

its scheduling and basic arithmetic knowledge to figure out what makes the proposed job different from all the others, using features like the relations between jobs, the resources required by single actions, and the position of an action in a sequence.
(2c) Based upon this analysis, Sched-Soar memorizes *explicitly* those aspects of the situation that seem to be responsible for the supervisor's advice. We call this kind of chunk an *episodic* chunk. Episodic chunks implement search-control knowledge, specifying *what* has to be done and *when*. An example is: If two jobs are already scheduled *and* three operators suggesting three jobs to be scheduled next are proposed *and* job 1 has the shortest processing time compared to the other jobs on machine A, then give a high preference value to the operator to schedule job 1. This kind of memorizing is goal-driven, done by an operator, and would not arise from the ordinary chunking procedure without this deliberation.

If in subsequent trials a similar situation is encountered, then Sched-Soar will bring its memorized knowledge to bear. Because the memorized information is heuristic, positive as well as negative transfer can result. And because only explicit, specific rules are created, general declarative rule based behavior appears to arise slowly and erratically.
(3) The job is assigned to the machines and bookkeeping is performed.

## Results

Figure 1 shows the solution time of 4 series of 16 randomly created tasks that were solved by Sched-Soar. Neither the power function ($r^2 = 0.55$) nor a simple linear function ($r^2 = 0.53$) proves a good fit to these data. Figure 2 shows that when averaged over series the data fit a simple power function well ($T = 274.0 \times N^{-0.3}$ with $r^2 = 0.95$).

A closer look at the processing times shows that the variance in the individual trials comes from two main sources: the negative transfer of chunked knowledge and the number of requests for advice. Negative transfer results, when an episodic chunk, built during solving a previous task, suggests an action in a situation that appears similar to the prior one, but which requires a different action. If this occurs, the situation has to be evaluated again to find the proper schedule-element, and, finally, if there is no suitable knowledge, the model still has to ask for advice. This explains why we found in the model's performance that additional requests for advice are often preceded by one or more instances of negative transfer. Both negative transfer and asking for advice directly lead to more deliberation efforts as measured in decision-cycles.

## Evaluation of the model

The model was evaluated (1) by investigating how many of the empirical constraints were met, and (2) by comparing the model results to empirical data in a further study.

(1) The results of the model show that the empirical constraints shown in Table 1 are met in general. (a) Solving the task requires 151 DCs, averaged over all trials and runs. The Soar architecture specifies the rate of decision cycles within half an order of magnitude with a center point at ten per second. This only constrains the time per cycle to be between 30 ms to 300 ms (Newell, 1990). The model performs slightly faster than the mean expected rate of 100 ms/DC, but at 147 ms/cycle it is well within the theoretical bounds.

Table 1: Constraints met and not met

| | |
|---|---|
| Total processing time | met |
| General speed up effect | not met |
| Improvement of solutions | undecidable |
| Suboptimal behavior persists | met |
| Algorithm not learned | met |

(b) The speed-up of the model is much greater than the subjects' — 57 % (from 270 DCs in the first trial to 118 in trial 16) compared to 22%. (c) The improvement in correctness cannot be decided yet, since Sched-Soar was initially programmed to always use advice to produce correct solutions. (d) The model did not discover the general algorithm. (e) Sched-Soar's behavior is suboptimal after 16 trials (and negative transfer might still occur in later trials).

(2) The model results can be considered theoretical predictions about subjects behavior in learning to solve the task. Some of these predictions were further evaluated in an additional empirical study because the task in the study forming the initial empirical constraints (Krems & Nerb, 1992) was not exactly the same as that solved by the model (in the first study feedback was given after a complete solution, whereas Sched-Soar is advised immediately after every single scheduling level decision). Therefore, a second experiment was conducted where the exact same task given to the model was given to 14 subjects. Subjects were instructed to separate their decisions based on knowledge from those based on guesses: They were requested to ask for advice when they did not know what to do. Each subject solved a total of 18 different scheduling problems.

These subjects' learning rate is shown in Figure 3. We found that a power function ($T = 109.6s \times N^{-0.38}$) accounts best for the averaged data ($r^2 = 0.82$, compared to 0.71 and 0.73 for linear or exponential fits). The average processing times for trial 1 to 18 vary between 99.4 and 36.3 s. Like many cognitive models (e.g., Peck & John, 1992), Sched-Soar performs the task more efficiently than subjects do, predicting values between 270 and 116 decision cycles. That means one has to assume 369 or 313 ms/DC, which is slightly above the region defined by Newell (1990). The learning rate (power law coefficient) of the subjects is approximately 26% higher than the learning rate of the model (-0.3 versus -0.38). In general that means that the task is easier for the

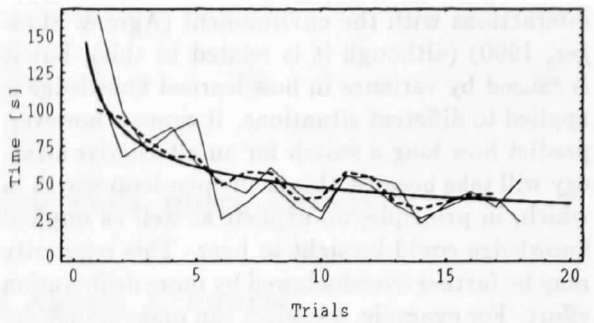


Figure 3: Processing time from trial 1 to 18 for 2 individuals (thin solid lines), the average solution time of all subjects (dashed line), and a power law fit to the average.

model and its performance improves more quickly (but will also stop improving more quickly). If this time constraint is taken seriously, future extensions to Sched-Soar should include more of the tasks that subjects must perform that Sched-Soar has performed for it, such as waiting for advice to be generated, reading the jobs off the screen, and typing in the schedule. Another explanation for the differences between the model's and the subjects' behavior might be based on the assumption that the learning mechanisms of Soar and human subjects are qualitatively similar but that there are quantitative differences. The model's behavior might be more efficient because it learns on every opportunity, whereas human subjects learning may be more specific than we propose here, or even probabilistic. We also found a correlation of $r = 0.46$ between subjects' requests for advise, due to lack of knowledge or wrong decisions, and processing time. This corresponds with the finding about negative transfer of chunked knowledge of the model, suggesting that the striking similarities in Figure 1 and 3 are both produced by the same mechanism. But this must be examined on a finer level before we note them as equivalent.

## Conclusions

A major claim of this analysis is that the power law of practice (Newell, 1990) will appear when performing the kind of scheduling task used in these studies, but only when the data are averaged over subjects or trials. In this task, the cause of the variance is shown clearly not to be noise in the measurement process or variance in the processing rate of the underlying cognitive architecture (which might have been proposed for simpler, more perceptual tasks), or through improved

interactions with the environment (Agre & Shrager, 1990) (although it is related to this), but it is caused by variance in how learned knowledge is applied to different situations. It cannot, however, predict how long a search for an alternative strategy will take because this is an open-loop search in which, in principle, all explicit as well as implicit knowledge could brought to bear. This regularity may be further overshadowed by more deliberation effort. For example, a subject can make simple decisions to ask for advice, or start more elaborate lookahead search to an arbitrary depth constrained only by their working memory and, of course, their motivation. Stripping away this time or replacing it with a constant factor should also yield a power law function on the empirical side, but only when averaged. A more fine-grained analysis (Agre & Shrager, 1990; Ritter, 1993) is required (and is already planned) to look at the firing of each episodic chunk that lead to negative transfer, comparing this with each subject's behavior.

On the other hand, further empirical work will be necessary to answer some of the questions posed by the model. For example, when will a strategy change take place, and will the results be of a local or global nature?

This is one of the first problem solving models to use episodic memory based on learning through reflection to learn a task in a cognitively plausible manner. It also shows how both the acquisition and storage of general rules in memory can be modelled by the Soar-architecture through acquisition of specific, context dependent rules (in contrast to VanLehn's (1991, p. 38) account). As this model is further developed, it should prove useful for explaining other aspects of scheduling behavior (e.g., the effect of further kinds of feedback on rule acquisition) and provide a possible new approach to constraint-based planning.

## Acknowledgments

We thank Aladin Akyurek, The Netherlands, for his help in getting Soar running in Regensburg. The paper was improved with comments from Peter Bibby and two anonymous reviewers.

## References

Agre, P.E., and Shrager, J. 1990. Routine evolution as the microgenetic basis of skill acquisition. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, 694-701. Hillsdale, N.J.: LEA.

Fox, M., Sadeh, N., and Baykan, C. 1989. Constrained Heuristic Search. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 20-25. Menlo Park, CA: International Joint Conferences on Artificial Intelligence, Inc.

Graves, S.C. 1981. A review of production scheduling. *Operations Research*, 29:646-675.

Johnson, S.M. 1954. Optimal two and three-stage production schedules with set up times included. Naval Research Logistics Quart, 1:61-68.

Krems, J., and Nerb, J. 1992. Kompetenzerwerb beim Lösen von Planungsproblemen: experimentelle Befunde und ein SOAR-Modell (Skill acquisition in solving scheduling problems: Experimental results and a Soar model). FORWISS-Report, FR-1992-001.

Minton, S. 1990. Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42:363-391.

Newell, A. 1990. *Unified theories of cognition.* Cambridge, MA: Harvard.

Peck, V.A., and John, B.E. 1992. Browser-Soar: A computational model of a highly interactive task, *CHI'92.* New York: ACM-Press, 165-172.

Prietula, M.J., Hsu, W.L., Steier, D.M., and Newell, A. 1993. Applying an architecture for general intelligence to reduce scheduling effort. *ORSA Journal of Computing.* Forthcoming.

Ritter, F.E. 1993. TBPA: A methodology and software environment for testing process models' sequential predictions with protocols. CMU-CS-93-101, School of Computer Science, CMU.

Rosenbloom, P.S., Laird, J.E., Newell, A., and McCarl, R. 1991. A preliminary analysis of the Soar architecture as a basis for general intelligence. *Artificial Intelligence*, 47: 289-325.

Sanderson, P.M. 1989. The human planning and scheduling role in advanced manufacturing systems: An emerging human factors domain. *Human Factors*, 31(6):635-666.

Smith, S., Ow, P.S., Potvin, J., Muscettola, N. and Matthys, D. 1990. An integrated framework for generating and revising factory schedules. *Journal of the Operations Research Society*, 41:539-552.

VanLehn, K. 1988. Toward a theory of impasse-driven learning. In Mandl, H., and Lesgold, A. (Eds.), *Learning issues for intelligent tutoring systems.* 19-41. New York, NY: Springer.

VanLehn, K. 1991. Rule acquisition events in the discovery of problem-solving strategies. *Cognitive Science*, 15:1-47.