1989

PERFORMANCE =/= BEHAVIOR: A STUDY IN THE FRAGILITY OF EXPERTISE

Brian R. Huguenard
Carnegie Mellon University

Michael J. Prietula
Carnegie Mellon University

F. Javier Lerch
Carnegie Mellon University

PERFORMANCE ≠ BEHAVIOR: A STUDY IN THE FRAGILITY OF EXPERTISE

Brian R. Huguenard
Michael J. Prietula
F. Javier Lerch
Graduate School of Industrial Administration
Carnegie Mellon University

ABSTRACT

The fragility of expertise is a known, but little understood, feature of expert reasoning. Essentially, *fragility* refers to the performance degradation of experts as task properties change. A study is presented in which the fragility of expertise in a complex, real-world task -- reactive scheduling -- is investigated. Six novices (students, trained in the task but with no experience in the domain) and three expert schedulers (ranging from six to 20 years of experience in the domain) each completed six reactive scheduling tasks varying in difficulty. All subjects were run individually and their protocols (verbal and action) were recorded on video-tape. Simple modifications to the task environment were sufficient to degrade the *performance* of the experts, sometimes to the level of the novices. However, an analysis of the *behavior* of the subjects suggests that a problem space characterization of fragility can explain how that degradation occurred. The behavior captured in the video-tapes (both verbal utterances and physical actions) show that, in this task, the primary source of degradation was the inappropriate formation of problem space components. That is, experts were "stuck in the wrong problem space." Specifically, the experts would use inadequate search control knowledge while traversing problem spaces and/or repeatedly attempt to implement operators or types of search control knowledge that were not allowed in the experimental task, but were quite valid in the real task setting. We conclude by discussing the concept of expert fragility and how it should be taken into account when designing systems based on the construct of expertise: expert systems.

1. INTRODUCTION

Human expertise is a critical resource and will become increasingly so as society's tools and techniques for acquisition, creation, distribution, control, and management of information become more knowledge intensive. One claim, echoed by developers of expert systems, is that human expertise is *fragile* -- changes in the problem or problem context may result in dramatic performance degradations (Brown and Campione 1984; Reed, Ernst and Banerji 1974). As expert system development becomes more and more widespread, developers should be wary of the possibility that their systems will inherit this flaw and its ramifications. Although the concept of fragility has an intuitive appeal, few studies have been conducted to explicate the nature of this phenomena, that is, few studies have attempted to discover where and how such fragility is manifest. With this study we are beginning to explicate the nature of expert fragility. The plan of the reported study is to compare how experts and novices perform on a task that has been modified to degrade the performance of the expert to that of the novice, but still permits the behavior of the expert and novice to be investigated in detail.

When investigating expert behavior, it is important to make a distinction between performance and behavior. *Performance* refers to the level of accomplishment achieved in the context of the task; *behavior* reflects the nature of the cognitive processes which lead to performance. Even though *performance* levels of an expert can be manipulated to approach that of a novice, the *behavior* of the expert should still be radically different. The expert performance diminishes, but the foundations of expert behavior, the relevant knowledge and its organizations, remain.

Our interpretation of the underlying behavioral processes involved is based on information processing psychology, which takes the view that humans are members of a class of general purpose symbol manipulators -- physical symbol systems (Newell 1980a, 1969; Newell and Simon 1972).[1] A component of this theory, and central to our interpretation of fragility, is the notion that all problem solving behavior can be explained in terms of problem spaces and operators (Newell 1980b). We propose that performance degradations suggesting expert fragility are based on specific and identifiable difficulties with problem spaces and operators.

In this paper, we first briefly describe the relevant research which investigates the fragility of knowledge in general. Second, we present the relevant theoretical background by introducing the concepts of expertise, task environments, and problem spaces. Third, we propose a view of task fragility based on information processing psychology which

embodies the concepts of problem spaces and operators. Fourth, we describe a study which explores our notion of fragility by examining how expert and novice schedulers perform a series of complex, reactive scheduling tasks.[2] Fifth, we examine and discuss the results of the study. We conclude the paper by exploring the implications for the design of expert systems suggested by the findings.

2. PRIOR RESEARCH ON FRAGILITY

In general, a problem solver's knowledge is organized around key features of the problem domain, and if any of these key features are altered, then the problem solver's performance may be negatively affected. Several researchers have demonstrated how the efficiency with which people perform tasks can be effected by the task representations used during task performance. For example, Adelson (1984), in a study of programming tasks, demonstrated how expert programmers tend to develop abstract representations of programs, while novices tend to develop representations based on surface level features of programs. Adelson showed how these differences in representations can lead to novices outperforming experts for certain programming tasks. She explained this seemingly counterintuitive result by noting that the novices' representations were well suited for tasks requiring knowledge of concrete, surface level program features, while the experts' representations had lost such details in favor of more abstract, conceptual knowledge.

Douglas and Moran (1983) studied subjects learning to use a text editor. These subjects were using an analogy to their knowledge of typewriters in order to understand the text editor, and this use of analogy caused them to try to implement inappropriate actions (such as trying to use the space bar to move the cursor past a word). This is an example of a mismatch between a mental representation and the actual task; in this case, the result was the attempted use of operations which, while appropriate in the domain of the mental representation, were incorrect in the actual task domain.

Johnson et al. (1981) investigated how errors in medical diagnostic reasoning could be attributed to deficiencies in knowledge about specific disease knowledge. In this series of studies they compared how expert, intermediate and novice diagnosticians solved particularly difficult problems. By examining how a computer simulation also solved similar problems, they successfully predicted where and how diagnosticians would *fail* in diagnosing diseases when particular types of evidence were degraded. The nature of the diagnostic failures were of two types. The first type simply involved "not thinking of the correct answer." This failure to generate a (known) disease was found to be based on the inability of the subjects to correctly interpret the meaning of a small number of critical data cues. The second type of error was based on a proposed, but inadequate, model of disease hypothesis. Specifically, expectations for an alternative disease were incorrectly judged as "better matched" than those expectations for the correct disease.

The studies of Adelson (1984), Douglas and Moran (1983), and Johnson et al. (1981) show that interactions between the task environment and a person's mental representation of the task can have negative impacts on task performance. The present study extends this type of research by demonstrating how representations and solution methods used by experts in a highly complex task (reactive scheduling in a production environment) can be rendered inefficient by slight changes in the task environment. However, we offer a view of fragility that goes beyond the present set of studies and argue that fragility results from specific and key failures of the problem solving knowledge and not from any type of general collapse of the expert's knowledge or the problem solving architecture.

3. EXPERTISE, TASK ENVIRONMENTS, AND PROBLEM SPACES

Expertise in a particular domain depends on the knowledge and properties characteristic of that domain (Newell and Simon 1972; Simon 1981). Expertise is based on what the expert knows and how that knowledge is used to solve problems. The particular characteristics of reasoning manifested by a given individual (or set of individuals) is a joint function of the properties which the tasks of the particular domain afford in the task environment and the nature of the individual's adaptation to them reflected in problem space formation. The nature of adaptation, then, is based on the formation of problem space components in response to demands of the task environment (Newell 1980b, 1969; Newell and Simon 1972).

A *task environment* refers to "an environment coupled with a goal, problem or task -- one for which the motivation of the subject is assumed" (Newell and Simon 1972, p. 55). It is the problem as presented to the problem solver and viewed by an "omniscient observer" (Simon 1978). In order to solve a problem, the problem solver must create an internalized representation of the task environment -- the problem space.

A *problem space* can be envisioned (abstractly) as a set of nodes representing various attainable knowledge states (i.e., that which the problem solver "knows" in the state), with one or more distinguished states representing the solution to the problem -- the goal. Problems are solved by *search* in the space; furthermore, this search is accomplished by a set of procedures comprising search control mechanisms. These search control mechanisms (i.e., search control knowledge) engage and monitor problem spaces, states, and operators. Operators are applied to a problem space state yielding a new state; this new state then becomes the focus of attention for the application of

operators, and the process continues until a satisfactory solution state is generated.

Search control knowledge is partial knowledge of how to proceed with the search in a problem space. When little is known about a particular (and complex) problem, then general (i.e., less task specific) methods are used for searching and representing the problem space -- these are the *weak methods* (Newell 1969) such as generate and test, progressive deepening, iterative deepening, best-first search, exhaustive maximization, analogy by implicit generalization and hill climbing. In problem solving situations which are knowledge-lean, weak methods may be the initial (and perhaps only) ones brought to bear. The behaviors these methods produce are correspondingly simple, as little task-relevant knowledge is available to exploit aspects of the task environment to reduce the search effort.

Experience and knowledge within a particular domain, however, lead to modification of both the problem space and the search control knowledge which allows more effective search (i.e., problem solving). These *strong methods* are based on components reflecting adaptation to the task in such a manner as to incorporate forms which improve the level of problem solving ability. By exploiting regularities in the task environment, such methods produce behavior which is correspondingly less simplistic, more specialized, and more appropriate for solving the problem at hand.

Multiple problem solvers working in the same domain will be likely to develop similar (though perhaps not identical) search control knowledge, since they work within the same task environment of operators, goals, and constraints. When faced with a common problem, problem solvers with similar search control knowledge would be likely to generate functionally equivalent operator sequences. On the other hand, problem solvers who are new to a domain do not have domain-specific search control knowledge to guide their choice of operators, and so a group of novice problem solvers faced with the same problem will be likely to generate a diverse, and functionally inequivalent, set of operator sequences.

The foundation of expertise is thus based on adaptations to the task environment within the information processing constraints of human reasoning. The nature of these adaptations permits these constraints to be functionally exceeded; that is, the adaptation entails the development of problem spaces, search control knowledge and operators which are *powerful*. Given this characterization of expertise and problem solving behavior, we should expect that for the tasks of this study, expert problem solving paths exhibit more directed and efficient problem space search than those for novices and that, given the same task, the expert group will generate less diverse operator sequences than the novice group.

4. THE STUDY

4.1 Subjects

Six novices (identified as N1 through N6) and three experts (identified as E1 through E3) participated in the study. The novices were undergraduates recruited from a small, private university, and the experts were highly experienced schedulers employed by a production facility from which the experimental scheduling task was drawn. The three experts differed in time spent performing the real world scheduling task. The most experienced expert had 20 years of scheduling experience; the other two experts each had from six to eight years of scheduling experience in addition to over ten years of experience in the supervision of scheduling.

4.2 Materials

The basic situation for the problem solvers involved scheduling jobs for three machines running in parallel (continuous process, no rework). Each type of job had a set of characteristics associated with it as did each of the three machines. Constraints on scheduling (e.g., what job can run on what machine, which jobs can be run together on the same machine) were defined based on these properties. A representation of a production schedule format resembling a Gantt Chart was developed on a metallic board, and rectangular magnetic cards were used to represent jobs. The magnetic cards were placed on the board allowing the subjects to manipulate the schedule by physically moving the cards. The schedule format on the board consisted of three parallel time lines reflecting scheduling slots for jobs on the three machines. The cards could be slid from one machine schedule-slot to another, or picked up and placed anywhere on the board. Each card had information identifying the type and characteristics of the job represented by it. Small, triangular magnetic tabs represented the presence of required setups between certain types of jobs (a setup is an equipment change to a machine and takes a fixed amount of time).

4.3 Procedure

The entire procedure took an average of two hours to complete for each subject and was divided into two main sessions: the introductory session and the actual task performance session. All subjects were asked to verbalize their thoughts while executing the assigned tasks. These verbalizations, along with the subjects' actions, were recorded on video-tape.

Introductory Session. The purpose of the introductory session (30 to 45 minutes) was to acquaint the subject with the task domain and give the subject a chance to practice tasks similar to those in the actual experiment. Before the actual experiment began, each subject was given a tutorial

covering the basics of the production process, the representation scheme of the cards, the constraints involved in scheduling, and the scheduling objectives. The scheduling objectives presented to the subject were the following:

1) avoid violating *constraints*,
2) schedule the jobs in order of *priority*, and
3) minimize the number of *setups*.

Constraints reflect physical realities whose violation invalidates a schedule (e.g., scheduling at a volume than is not achievable by a machine). Feasible schedules are those that do not have any constraint violations. The *priority* of a job refers to the relative demand and supply for each job type; the subject was provided with a priority document which could be used to rank jobs' priorities relative to each other and was available throughout the session. The subjects were told to take priority into account (schedule jobs with high demand and low supply before those with low demand and high supply) when making changes in the schedule, and they were told that the concept of priority was *secondary* in importance to that of feasibility. *Setups* involve changes that have to be made to the machines between certain types of jobs.

The subjects were given two practice tasks (two insertions: one "easy," one "difficult," as described under Experimental Tasks) and were asked to verbalize their thoughts during task performance.

Experimental Tasks. After the introductory session, each subject was presented with a sequence of three different two-week schedules. For each schedule, two tasks were given, for a total of six tasks. A time limit of 30 minutes was imposed on each task. There were three different types of tasks (insertion, postponement, and shutdown), and for each task type there were two levels of difficulty (easy, difficult).

For an *insertion task*, the subject was presented with two cards representing new jobs to be scheduled in the first week of the two week schedule. In *postponement tasks*, the subject was told that some raw materials required for a particular job would not be arriving on time and therefore the job would have to be re-scheduled after a specified time. In a *shutdown task*, the subject was told that a particular machine would be inoperative during a given eight hour span, and the job(s) that was(were) scheduled during that time would have to be rescheduled. An easy task was defined as one which involved a job that would not, as a result of its movement within the schedule, introduce many constraint violations. On the other hand, a difficult task was defined as one which involved a job which, if moved, would be likely to introduce constraint violations. These added constraint violations then would have to be resolved, adding to the total number of required schedule manipulations, thus increasing the difficulty of this type of task.

Data Collection. Subjects were videotaped during task performance, capturing both their actions and their verbalizations. Concurrent, neutral-probing verbalizations were used (Ericsson and Simon 1980). To aid in reconstruction of the subjects' moves, the experimenter took notes during task performance, identifying the part numbers of the cards being moved as well as the origin and destination of the moves.

5. RESULTS AND DISCUSSION

The *performance* of the experts in the experimental tasks was similar to that of the novices in terms of quantitative measurements such as the time needed to complete the tasks, the number of moves executed and the quality of the end product (the schedule). On the other hand, the sequence of the moves and the verbal protocols elicited during problem solving (i.e., *behavior*) indicate very different problem solving paths between the two groups. In this section, we first analyze both performance and behavior differences and similarities between experts and novices. Then in the next section, we explain the concept of fragility of expertise and its causes using a problem space characterization of the subjects' behavior.

5.1 Performance

Physical movements and timing data were transcribed from the videotapes. Movements were classified into four general categories (in these descriptions the term "object" can refer to either a single card or a set of cards representing jobs on the three different machines):

- **Move** Remove an object from the board and place it at an unoccupied location (this operation can cross machines).
- **Swap** Interchange the location of two objects, all in one operation.
- **Shift** Slide an object along a machine schedule (this operation can not cross machines).
- **Remove** Remove an object from the board (this generally was invoked to temporarily hold an object for future rescheduling).

The dependent variables measured were total solution time, total number of moves, number of each type of move, and schedule quality. Schedule quality was measured according to the number of constraint violations and the number of added setups.

The second and third columns of Table 1 give a comparison of the performance of the expert and novice groups for the first four tasks. Subject fatigue during the fifth and sixth tasks rendered the validity of those task results

Table 1. Overview of Subject Performance for Tasks 1, 2, 3, and 4

	Task Characteristics	Performance		Evidence of	
		Experts (n = 3)	Novices (n = 6)	Expertise	Fragility
Task 1	• Type: shutdown. • Difficulty: easy.	• avg solution time: 1.04 min • move sequences: all identical • final solutions: all identical • solution quality: no added setups	• avg solution time: 3.63 min • move sequences: all different • final solutions: 2 same as experts; 4 other diverse solutions • solution quality: 1 solution infeasible; 2 others with at least 1 added setup	• Experts' solution times significantly shorter than novices'. • Experts' move sequences and final solutions identical. • All experts gave verbal indications of prior knowledge of solution.	None.
Task 2	• Type: postponement. • Difficulty: difficult.	• avg solution time: 15.49 • move sequences: initial sequences identical, followed by divergent paths • final solutions: all different • solution quality: 1 solution infeasible, other 2 had no added setups	• avg solution time: 17.15 • move sequences: all different • final solutions: all different • solution quality: 4 out of 6 solutions infeasible, other 2 solutions had 2 and 0 added setups	• Time for experts to execute initial move sequence very short. • Experts' initial move sequences identical. • All experts gave verbal indications of prior knowledge of initial sequence.	• Persistent proposal of invalid operators. • Interference of task constraints and real-world constraints. • Misinterpretation of problem features.
Task 3	• Type: insertion. • Difficulty: easy (easiest of all tasks).	• avg solution time: 1.28 • move sequences: all identical • final solutions: all identical • solution quality: all solutions had 1 added setup (unavoidable)	• avg solution time: 5.75 • move sequences: 4 out of 6 sequences identical • final solutions: 5 out of 6 solutions identical • solution quality: all solutions had 1 added setup (unavoidable)	None (ceiling effect).	None.
Task 4	• Type: postponement. • Difficulty: easy.	• avg solution time: 6.08 • move sequences: 2 out of 3 sequences identical • final solutions: 2 out of 3 solutions identical • solution quality: 2 solutions had no added setups; other solution had 2	• avg solution time: 4.51 • move sequences: all different • final solutions: all different • solution quality: 1 solution infeasible, 3 of others had 1 added setup, remaining 2 solutions had no added setups	• 2 of 3 Experts' move sequences and final solutions identical (other Expert started with a similar solution, but then diverged from the others).	• Interference of task constraints and real-world constraints. • Misinterpretation of problem features.

questionable, so those tasks are not included in this analysis. For each group/task combination, four performance criteria are considered in Table 1: the group average for time to complete the task, the within-group similarity of move sequences, the within-group similarity of final solutions, and the quality of final solutions generated by members of the group.

Except for the first task, there are no statistically significant differences between average solution times for the two groups of subjects (Table 2). Experts are faster than novices at solving only the first task ($t(7) = 2.54$, $p < .05$), but there are no other significant differences between the two groups for any other measurement. There are no significant differences between number of moves for experts and novices for each task, nor are there significant differences between the two groups for move breakdowns by type of move (Tables 3 and 4). Finally, there are no significant differences between experts and novices in terms of quality of their schedules, when quality is measured in terms of the number of constraint violations and the number of added setups (One-tail Mann-Whitney tests, $p > .05$).

5.2 Behavior

Subject behavior was analyzed according to two dimensions: 1) the nature of their search control knowledge, and 2) the between-subject similarity of operator (move) sequences generated during problem solving. Although these two dimensions are not totally independent of each other, this type of analysis helps to explicate the differences between expert and novice behavior. The fourth column of Table 1 gives a concise listing of the evidence from the first four tasks that supports our expectations for expert behavior (given the same task, expert problem solving paths should exhibit more directed and efficient problem space search than those for novices, and the expert group should generate less diverse operator sequences than the novice group), while the following two sections discuss the evidence in more detail.

Search Control Knowledge. Expert and novice behaviors were analyzed for the first four tasks (an easy shutdown, a difficult postponement, an easy insertion, and an easy postponement). In the first task, experts were faster than

Table 2. Average Total Solution Times

Problem	Group: Novices	Group: Experts	t(7)
1	3.63	1.04	2.54*
2	17.15	15.49	0.27
3	5.75	1.28	0.93
4	4.51	6.08	- 0.57
5	14.62	22.27	- 1.63
6	12.46	13.04	- 0.07

* $p < .05$

Table 3. Average Total Number of Moves

Problem	Group: Novices	Group: Experts	t(7)*
1	4.33	2.00	0.73
2	32.17	22.67	0.67
3	6.33	1.00	0.98
4	5.83	9.00	- 0.49
5	17.67	21.00	- 0.32
6	23.67	22.00	0.08

* all t-values not significant

Table 4. Breakdown of Total Number of Moves by Move Type

Move	Group: Novices	Group: Experts
Move	38.33%	43.35%
Shift	24.44%	25.75%
Swap	15.00%	13.30%
Remove	22.22%	17.60%

$X^2 (3) = 3.13, p > .10$

the novices in solving the problem. The analysis of their behavior shows that experts used previously acquired search control knowledge to select a simple, but optimal, strategy for this task (strong method) while novices executed a quite general and deliberate search procedure (weak method). In the second task, expert performance was similar to novice performance, but expert behavior was highly dissimilar from novice behavior. In the third task, both performance and behavior of the two groups were indistinguishable. This was most likely due to ceiling effects since the solution to this task was obvious, regardless of prior knowledge of the domain. The third task results will not be discussed further. In the fourth task, expert performance was similar to novice performance, but as in the second task, expert behavior was dissimilar from novice behavior.

Task 1 -- Easy Shutdown. Figure 1 shows an abstract representation of the task given to the subjects. Subjects were told that an unexpected shutdown had to be scheduled for a given machine at a given time. This event was simulated by placing a "shutdown card" over a portion of the initial (and feasible) schedule. Step b in Figure 1 depicts this event by showing a shutdown card over job D. Therefore, job D is divided into two components: D1, which is before the shutdown, and D2, which is displaced by the shutdown. The subjects' task was to find some way to complete the processing of job D by either rescheduling D2 or by rescheduling the complete job D.

An efficient solution to this task, as shown in step c of Figure 1, is based on a strategy encompassing a series of moves to shift job E to the right (i.e., delaying its processing on the machine), thereby creating space (i.e., a time slot) after the shutdown to insert job D2 and complete the processing of that job. This strategy, called SHIFT-AND-COMPLETE, does not add additional setups, because the machine is still appropriately configured for job D after the shutdown. In addition, the solution generated from the SHIFT-AND-COMPLETE strategy does not upset the basic schedule of jobs on the machine, thereby maintaining the priority ranking of the jobs. Finally, the SHIFT-AND-COMPLETE strategy solution does not violate any constraints.

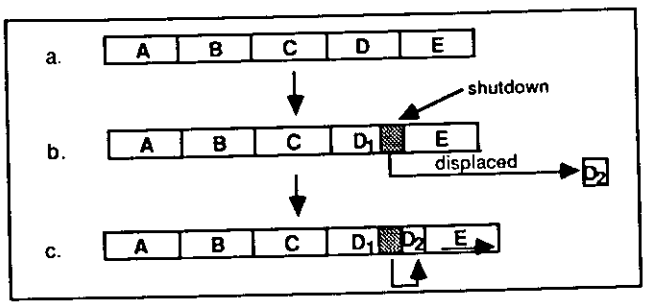

Figure 1. Move Sequence for "Shift and Complete" Strategy (Solution to Task 1)

Upon being presented with this task, all of the experts immediately applied the SHIFT-AND-COMPLETE strategy and generated the efficient solution. SHIFT-AND-COMPLETE requires only two moves, one Shift and one Move, and all of the experts performed these two moves (i.e., implemented the SHIFT-AND-COMPLETE strategy in the same way). Total solution times (in minutes) for the experts were 1.15, 1.05, and 0.93. Expert E1's protocol is representative of all the experts for Task 1, and shows how the expert's search control knowledge successfully guided the behavior in this problem. The annotated protocol below suggests that E1 had an efficient strategy available for this situation and applied it without further deliberation. Since this strategy was allowable and efficient in the experimental task environment, expert E1 was able to apply this knowledge without incurring conflict between this prior knowledge and the present task.

Protocol E1, Task 1 (Easy Shutdown)

[1:07] The question comes to mind...when you say you have a shutdown for maintenance...we have the option of shutting down [the machine] but retaining our present complement of irons in position [i.e., does not incur a setup for the machine].
[1:24] So what we can do is utilize this remaining time in the week, defer this eight hours, and go right back to, once the shutdown is completed, the same part, and complete the run.

The novices (subjects N1 through N6) had a variety of solutions to Task 1, ranging from infeasible solutions to the efficient one arising from the SHIFT-AND-COMPLETE approach. The two novices (subjects N4 and N5) who reached the solution generated by the SHIFT-AND-COMPLETE strategy did so by deliberate search and explored several alternative states and did not immediately apply the SHIFT-AND-COMPLETE strategy to generate the right solution as did the experts. The total number of moves for the novices ranged from one to 15 while the total solution time ranged from 0.70 to 5.35 minutes.

Novice N4's protocol depicts the generation of the efficient solution (via the SHIFT-AND-COMPLETE sequence of moves) through deliberate search. The protocol suggests that N4 did not have a preconceived SHIFT-AND-COMPLETE strategy available for Task 1.

Protocol N4, Task 1 (Easy Shutdown)

[N4 first notices a simple but inferior solution, and then decides to search for a better solution]:
[1:52] Now if I just simply put this card here...I'd need another setup, and that's something I'd like to avoid.
[Novice N4 then considers another solution (but not the optimal one)]:
[2:08] I'm wondering whether I can make up this 4 hours.
[2:42] This right here looks pretty good.
[5:35] The other option is to have a net difference of negative 4 hours.
[After considering several possible solutions, N4 finds the SHIFT-AND-COMPLETE solution]:
[6:10] It's just occurred to me that I should just move this down...I should just finish the other part after the shutdown.

Novice N5, the other novice who eventually generated the efficient SHIFT-AND-COMPLETE solution, shows through this protocol segment that he also did not have a preconceived SHIFT-AND-COMPLETE strategy available.

Protocol N5, Task 1 (Easy Shutdown)

[1:41] I suppose I'd rather have...run this four hours and keep an extra four hours on Saturday afternoon...or I could run this, all twelve hours on a different machine, I can't do that...or I could simply find a suitable place for this.

In summary, all experts immediately invoked the same version of the SHIFT-AND-COMPLETE strategy and arrived at the efficient solution with no deliberate search involved. This behavior is responsible for the commonly (but not universally) encountered expert performance of a substantially *shorter* time required to solve a problem. It is likely that experts did not outperform novices in the other quantitative measurements because of ceiling effects produced by the simplicity of the task or by the inadequacy of the measurements. Independently of the performance results, the differences in problem solving behavior *between* the two groups confirm the results of previous expert-novice studies (e.g., Chi, Feltovich and Glaser 1981; Johnson et al. 1981; Simon and Simon 1978).

Task 2 -- Difficult Postponement. In this problem the subjects were told that a given job had to be postponed. The problem is classified as difficult because this postponement introduces several constraint violations among the machines. An efficient solution path to this task is shown in Figure 2. Step a of this figure shows a postponement of job D after time x. The solution is based on a strategy that involves removing job D, shifting the jobs between job D and time x forward (i.e., processing them earlier), and then placing job D in the schedule at time x, resulting in the state shown in Step c in Figure 2. This is referred to as the SHIFT-AND-MOVE strategy.

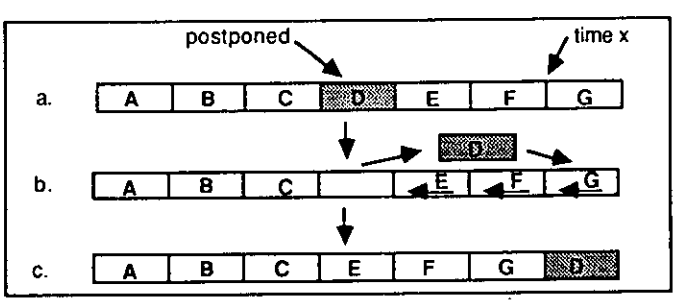

Figure 2. Move Sequence for "Shift and Move" Strategy (Solution to Task 2)

Though resolving most constraint violations, implementing the SHIFT-AND-MOVE strategy actually results in additional other constraint conflicts; however, these are minor and could be resolved in a multitude of ways, all involving swaps and shifts between machines. These constraint violation resolutions are not shown, as they

require only "patch up" operations to restore feasibility. The key point is that the SHIFT-AND-MOVE strategy results in a state which is very close to the solution and the minor scheduling adjustments needed to achieve the solution are a direct result of the application of the strategy. The resulting solution has no additional setups and retains most of the priority sequence of the jobs, while producing the postponed job as soon as possible.

The individual solutions for Task 2 are well fitted to be discussed in terms of problem behavior graphs (Newell and Simon 1972). These graphs represent progress through a problem space as a sequence of operator applications and resulting states. We will discuss them here in regard to the differences in search control knowledge between experts and novices.

Upon being presented with this task, all experts immediately began implementing a version of the SHIFT-AND-MOVE strategy and completed this phase within 1.57 to 2.77 minutes. After this strategy was implemented (i.e., realized and executed as a set of moves), the experts began searching for moves to restore feasibility. It was during this second phase that the experts took the most time (this phase took from 6.25 to 24.25 minutes) and had the most difficulty.

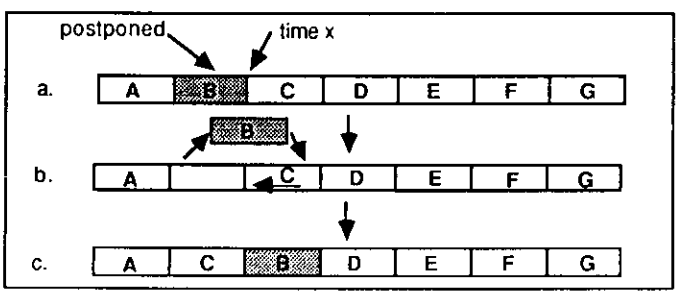

Figure 3. Move Sequence for "Swap" Strategy (Solution to Task 4)

Of the three expert solutions, two were feasible while the third was similar to the others but had one constraint violation (related to the second phase of solving the problem). Figure 3 shows E1's problem behavior graph. Each dot represents a different state, and the links represent operations. To follow the subject's behavior, start at the upper left corner, and follow the top line of dots horizontally to the right as far as possible. At the end of a line of dots, if this is not the final state, then go back (to the left) along the same line until the first vertical link is found, and go down this link to the first new horizontal line of dots. Now once again start going right as far as possible, and continue the process. The process of "backing up" to the left along the horizontal lines and then going down a vertical link replicates the subject's action of backtracking, where one or more unsuccessful operations are "undone," and a new sequence of operators is tried from a previously visited state. In Figure 3, the completion of the SHIFT-AND-MOVE strategy is noted by an annotated arrow. The protocol of E1 suggest once again that E1 had prior knowledge of this strategy as an appropriate one for the shutdown situation depicted in the task.

Protocol E1, Task 2 (Difficult Postponement)

[1:03] The easiest, and the thing I try to do and that I would recommend is that we would remove these from the schedule...temporarily
[1:20] And advance each item remaining in the schedule...assuming that all the other materials would be available and so forth...until that point is reached where these will be capable of running.

After implementing the SHIFT-AND-MOVE strategy (2:33 minutes), E1 noted some of the incurred constraint violations and began to resolve them. Note that no backtracking occurs until deep into the solution process (18:23). This backtracking will be analyzed in detail later in the discussion of fragility.

Figure 4 shows E2's problem behavior graph. In this graph there is an immediate backtracking (1:23) caused by E2 discovering a constraint violation generated by implementing a SHIFT-AND-MOVE strategy. However, E2 did not abandon the SHIFT-AND-MOVE strategy, but performed an exploratory swap of two jobs and continued with implementing the strategy. After completing the SHIFT-AND-MOVE strategy, E2 proceeded to resolve the violations and had only one other short instance of backtracking.

E3's problem behavior graph is given in Figure 5 and shows two instances of backtracking very early but, as with E2, E3 did not abandon the SHIFT-AND-MOVE strategy. After successfully implementing this operation, E3 continued with no backtracking to arrive at his final state.

Figures 6 through 11 show the problem behavior graphs for the six novices. Four of the novices (N1, N4, N5, and N6) began solving the problem by attempting a series of moves resembling the SHIFT-AND-MOVE strategy. Three of these four (N1, N4 and N6) abandoned it when they detected that constraint violations would occur, although N1 eventually returned to moves equivalent to that strategy. N5 did not abandon the SHIFT-AND-MOVE approach, but persisted and eventually successfully implemented it. The other two novices (N2 and N3) initially attempted other moves, but eventually incorporated moves comprising the SHIFT-AND-MOVE strategy in their behavior. Of the six novice solutions, only two were feasible.

The most obvious difference between the behavior graphs of the novices and the experts is the higher degree and earlier occurrence of backtracking in the novice graphs.

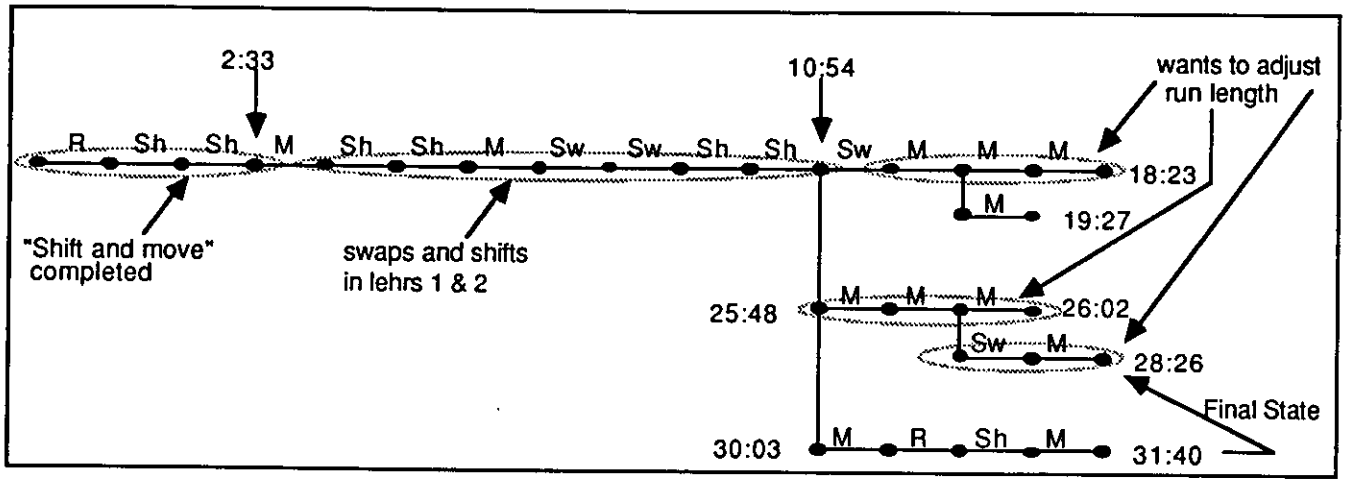

Figure 4. Problem Behavior Graph for Expert E1, Task 2

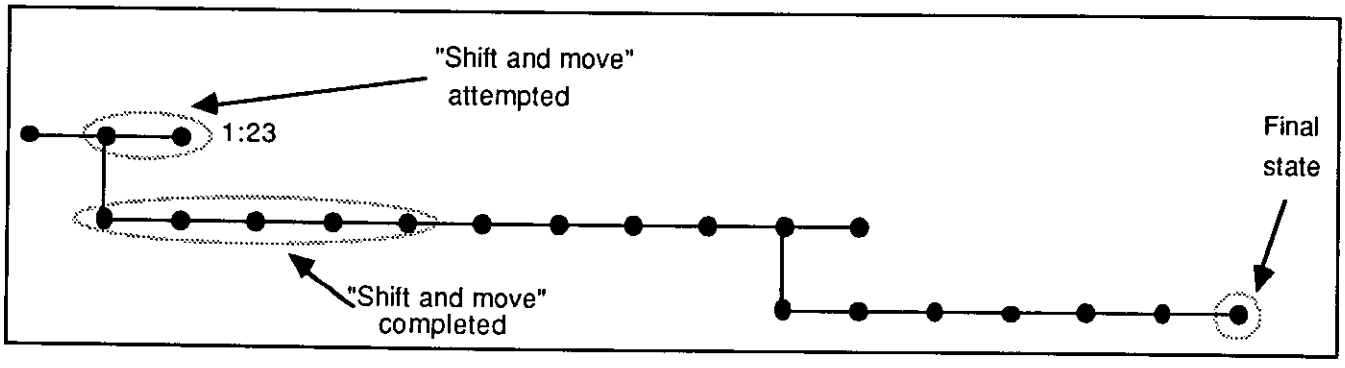

Figure 5. Problem Behavior Graph for Expert E2, Task 2

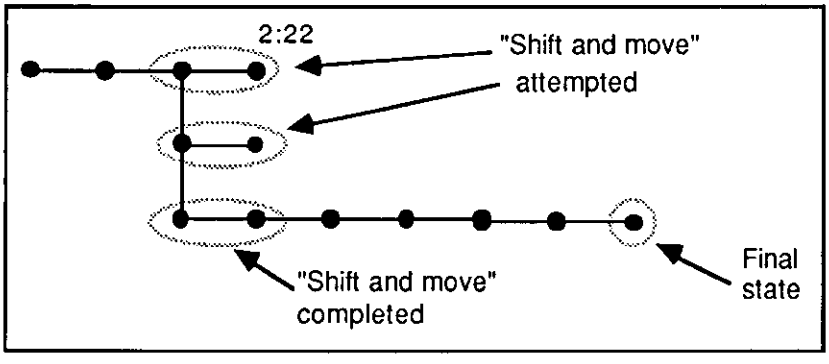

Figure 6. Problem Behavior Graph for Novice E3, Task 2

Novices frequently used shallow search; that is, they tried one or two moves from one state, then backtracked and tried a few other moves from the same state, and so on. This behavior is evidence that novices lacked an appropriate set of search control knowledge and their behavior was guided by weak (general) problem solving strategies. Having little domain-specific knowledge, they were forced to use general purpose problem-solving techniques to search for solutions. Through this search, some of the novices arrived at a series of operators that *comprised* the appropriate strategy, but the strategy itself was not available and could not direct search. Their sequence of moves was determined, again, by deliberate search.

On the other hand, experts exhibited little deliberate search or backtracking behavior and they all began with the same, directly invocable, strategy. After implementing this strategy, the experts had trouble correcting the incurred constraint violations. It was during this phase of correcting constraint violations that the experts began to

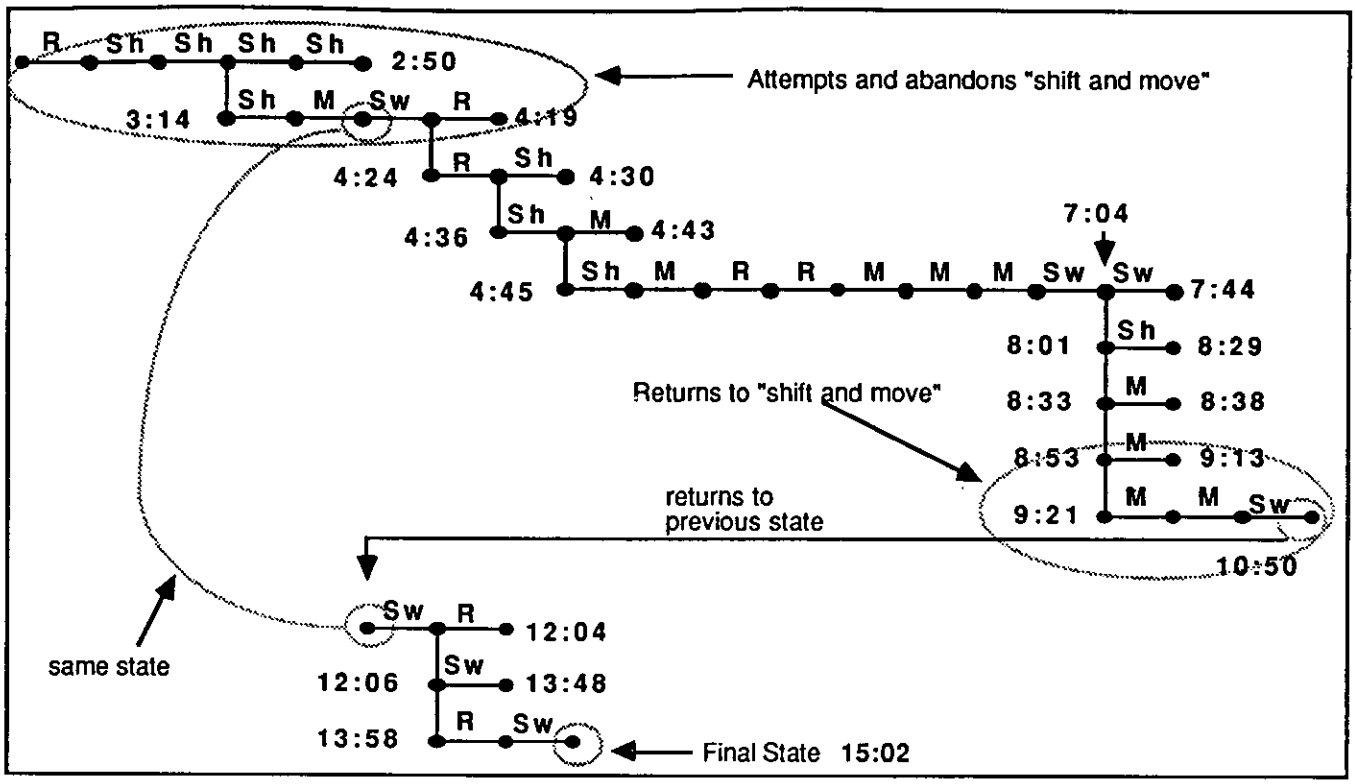

Figure 7. Problem Behavior Graph for Novice N1, Task 2

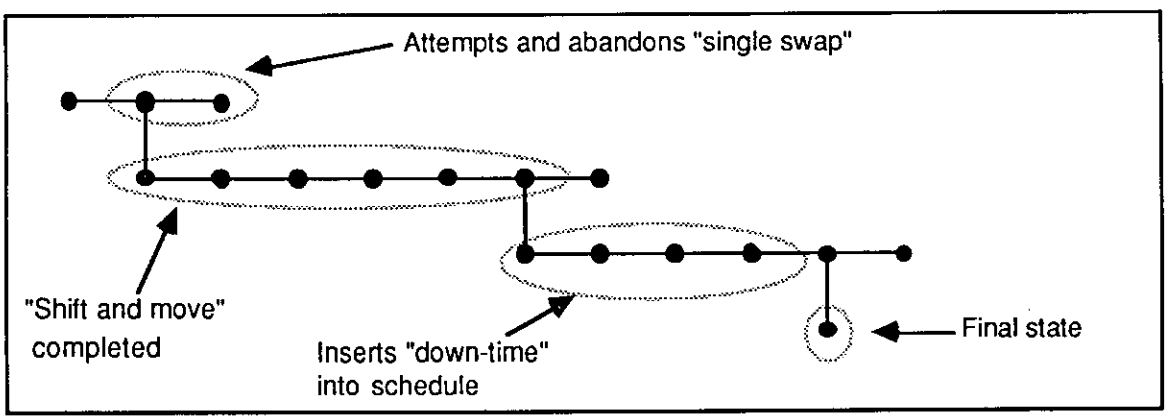

Figure 8. Problem Behavior Graph for Novice N2, Task 2

experience interference between their knowledge and the task (especially the most experienced expert, E1). This interference was the source of the degradation of expert performance as will be explained in the discussion of fragility.

Task 4 -- Easy Postponement. In this task, the subjects were again told that a given job had to be postponed. An efficient solution for this task is shown in Figure 12, and will be referred to as the SWAP strategy. In part a of Figure 12, job B has to be postponed until time x. Parts b and c show job B being swapped with job C, accomplishing the postponement with minimal disruption to the schedule, no added setups, and no constraint violations. For the task actually given to the subjects, job B was actually composed of three separate, but related, cards and the postponed job was represented by one of these cards. The SWAP strategy required all of these three cards to be treated as one unit during the swap with job C, making the actual task a bit more complex than the example given here.

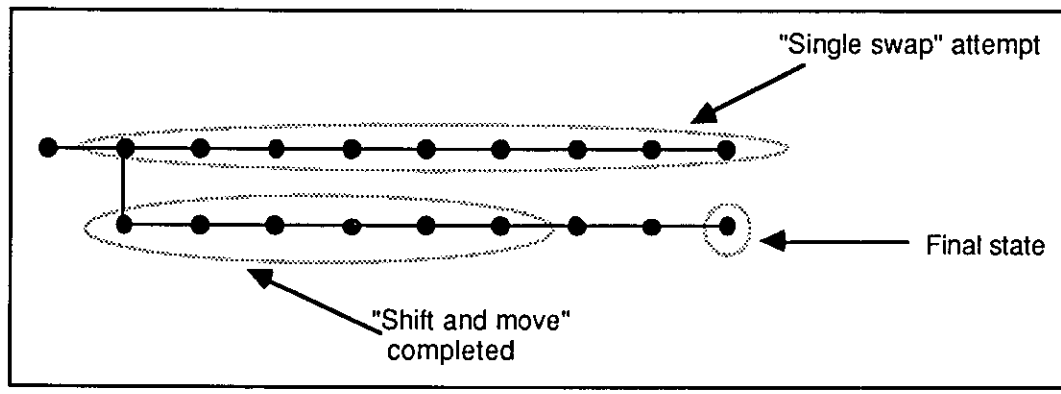

Figure 9. Problem Behavior Graph for Novice N3, Task 2

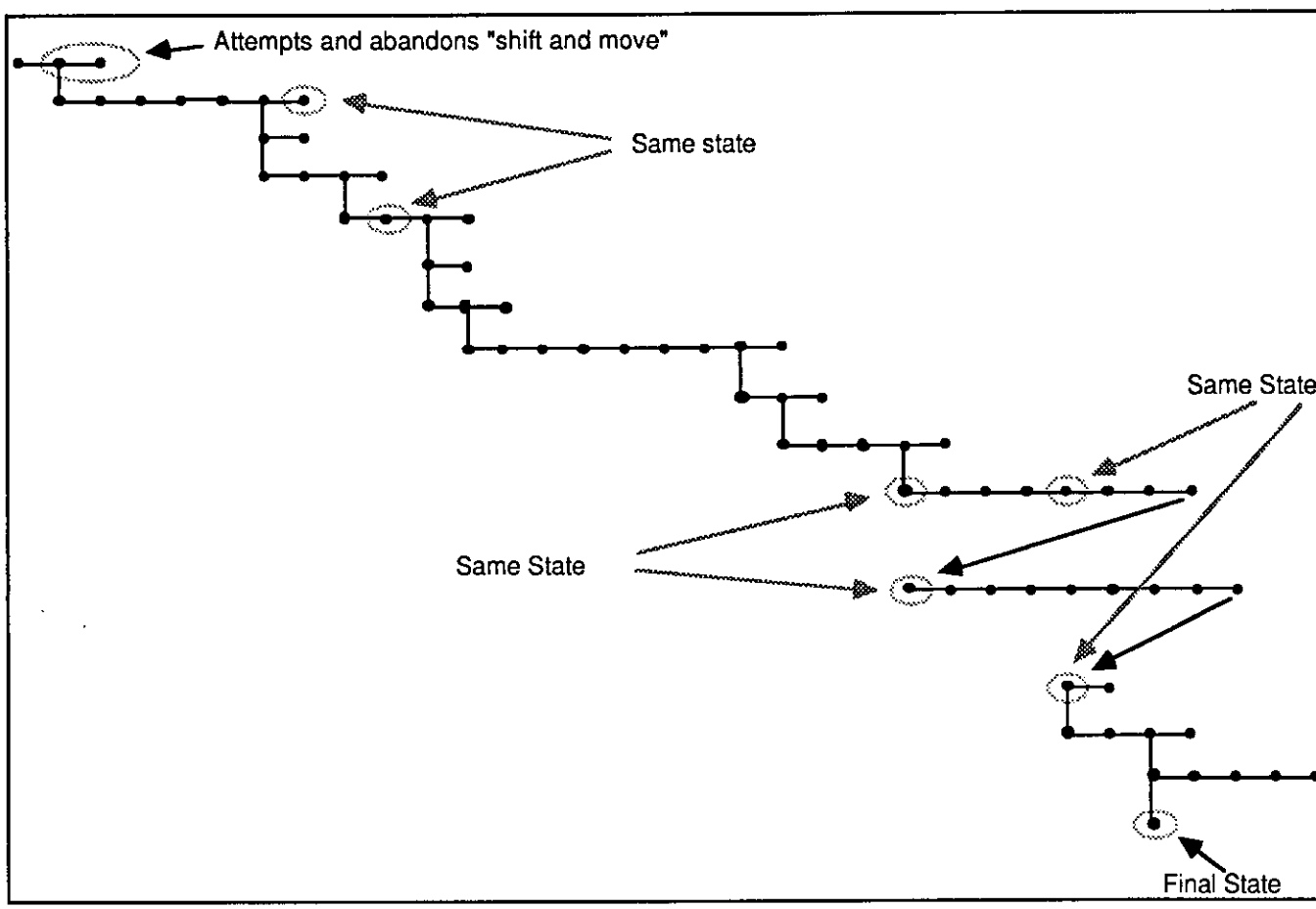

Figure 10. Problem Behavior Graph for Novice N4, Task 2

Upon being presented with this task, two of the experts (E2 and E3) quickly applied the SWAP strategy and generated the efficient solution. Total solution times for these two experts were 2.35 and 3.70 minutes. The other expert (E1) also quickly applied a strategy *similar* to the SWAP strategy (applied at time 1:07), but his version of the strategy did not include all of the appropriate parts. Due to this omission, E1's version of the SWAP strategy induced constraint violations, and forced him down a much longer, and quite different, solution path than that of the other two experts; his final solution had two added setups. Expert E2's protocol is representative of the two experts who generated the efficient solution, indicating once again that the experts' search control knowledge guided their behavior towards efficient solutions.

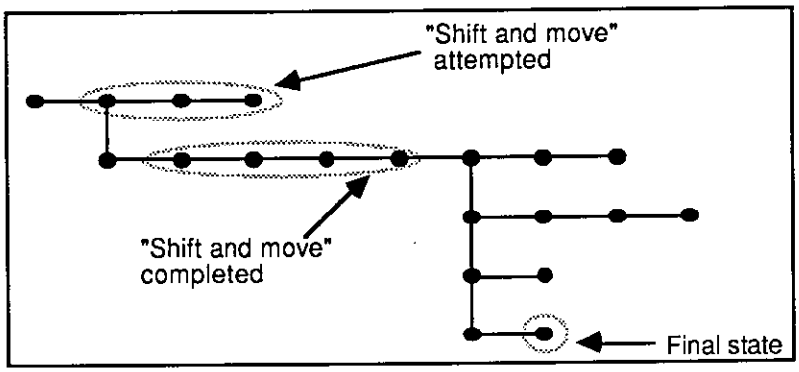

Figure 11. Problem Behavior Graph for Novice N6, Task 2

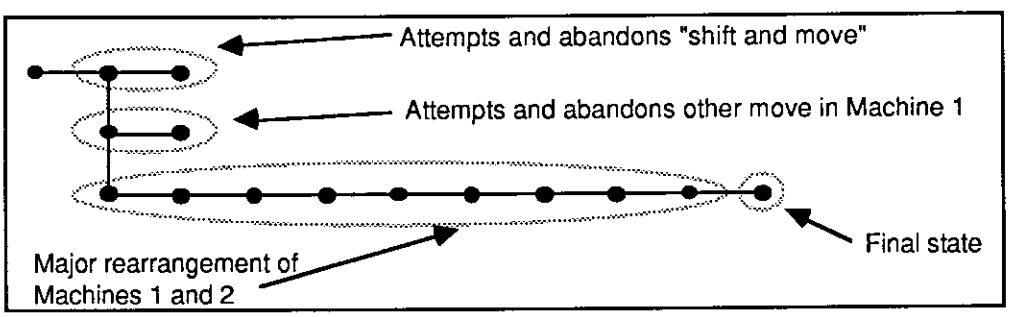

Figure 12. Problem Behavior Graph for Novice N6, Task 2

Protocol E2, Task 4 (Easy Postponement)

[1:14] Well, I know what I want to do, but I can't do it [actually, he can]. I want to take [part number] 52... [demonstrates the SWAP strategy]... that's what I want to do.

The novices had a variety of solutions, and no novice generated the SWAP strategy used by the experts. Of those solutions generated by the novices, one was infeasible, three had one added setup, and 2 had no added setups but were more disruptive to schedule priority than the SWAP strategy used by the experts.

In summary, two of the three experts invoked the same version of the SWAP strategy, with no deliberate search. The other expert started with a similar (but flawed) version of the SWAP strategy, and also implemented this strategy with no deliberate search, but then diverged from that solution path. In contrast, the novices generated a wide variety of solutions, four of which were inferior to the efficient solution generated by the experts. None of the novices demonstrated a pre-existing strategy to implement, and they therefore relied on general search methods to perform the task.

Operator Sequence. In terms of similarity of operator sequence, the differences in behavior *within* each group parallel expert-novice differences found by other researchers in previous studies. For example, McKeithen, Reitman, Rueter, and Hirtle (1981), Chase and Simon (1973), Prietula and March (1989), and Larkin, McDermott, Simon and Simon (1980) have all found in a variety of domains how experts tend to perform and behave consistently, while novices exhibit widely varying and inconsistent performance and behavior. Similar results were found in this study for the first, second, and fourth tasks.

Task 1 -- Easy Shutdown. In this task, the three experts exhibited similar behavior: recognize the situation, retrieve a previously learned strategy (i.e., a group of moves) and implement the strategy. That strategy (SHIFT-AND-COMPLETE) was implemented by each expert using exactly the same operator sequence. On the other hand, two of the novices arrived at the solution resulting from the SHIFT-AND-COMPLETE strategy, but only through the use of deliberate and general search while the other four novices arrived at inefficient solutions, also through the use of uninformed search (and one of these four solutions was not feasible). Each of the novices' solution paths employed different operator sequences, both in comparison to other novices and the experts.

Task 2 -- Difficult Postponement. In this task, while the overall operator sequence of the experts were not similar, all of the experts began their solutions with the same subsequence of operators (the SHIFT-AND-MOVE strategy).

In addition, the experts all implemented this sub-sequence quickly (within 1.57 to 2.77 minutes). After implementation of the SHIFT-AND-MOVE strategy, the experts' operator sequences began to diverge. On the other hand, the novices shared no sub-sequences of operators; each of their solutions were totally unique.

Task 4 -- Easy Postponement. In this task, two of the three experts immediately implemented the same sequence of operators (the SWAP strategy), both within 2.35 to 3.70 minutes. The other expert began with an operator sequence similar to the SWAP strategy, but then diverged and followed a much longer solution path. In contrast, all of the novices produced uniquely different operator sequences.

6. FRAGILITY OF EXPERTISE

We define fragility of expertise as *the degradation of expert performance as task properties change.* We propose that this degradation of problem solving performance is caused by the expert solving the problem in an inadequately configured problem space. The topology of the problem space is largely determined by the search control knowledge and the operators available to the problem solver. In the previous section, expert behavior was shown to be different from that of the novices because it exhibited domain-specific search control knowledge and a higher degree of similarity of operator sequences (as strategies). Therefore, the experts were engaging fundamentally different types of problem spaces than novices and actually behaving like experts, but their performance (especially in Task 2) was not better than the novices' performance. This degradation can be explained using the problem space characterization by showing that the problem space where the expert is solving the problem is inappropriate (i.e., the invocation of wrong knowledge) or by showing that the experts have adopted the strategies of the novices (i.e., the invocation of weak knowledge).

It is important to notice that performance degradation should only occur when the changes in task features render previously acquired knowledge to be inappropriate. The result of changing key task characteristics is that the expert either relies on general problem solving procedures or uses the inadequate knowledge given the task characteristics. In both cases performance is degraded either by extensive search in the problem space or by the misapplication of search control knowledge and/or operators. This section examines how seemingly small changes in task features produced by adapting the real-world task to the experimental setting resulted in the performance degradation of E1 (the most experienced subject) in Task 2 and Task 4.

E1 took the longest time to solve Task 2 and had the most serious difficulties among the experts after implementing the SHIFT-AND-MOVE strategy. These difficulties are first apparent through the comparison of problem behavior graphs: E1's graph shows the most backtracking among the experts. Three sources of performance degradation seem to dominate and each of them is the product of changing key features of the task: *persistent proposal* of invalid operators, *misinterpretation* of problem features, and *phantom constraints*. The last column of Table 1 gives a concise listing of the evidence for these sources of performance degradation for the first four tasks; this evidence is discussed below in more detail.

6.1 Persistent Proposal of Invalid Operators

The first source of degradation is that some operators available in the real-world are not present in the experimental task, although the experimental task is *similar enough* to the real-world task environment that the expert classifies some situations as ones in which the (invalid) operator would be appropriate. For example, in the real task the expert resolves the problem of multiple setups at different machines at the same time by what E1 calls "adjusting the run length." This strategy, ADJUST-RUN-LENGTH, is not available in the experimental task because the jobs are represented by magnetic cards of fixed length that can not be compressed or enlarged. The absence of this operator has two consequences. First, E1 finds it difficult to deal with setup constraints in the experimental task because this type of constraint can be resolved by using the ADJUST-RUN-LENGTH strategy in the real-world. Second, E1's knowledge of the real task triggers, on several occasions, the proposal of the ADJUST-RUN-LENGTH strategy even though E1 knows it is not possible to implement it. An example of the first difficulty is expressed in the following protocol.

Protocol E1, Task 2 (Difficult Postponement)

[6:28] Yeah, I say I've probably **negated** that, to the extent that the only reason you can't do that is because of....
The most delay you would need is 45 minutes or half an hour, to take your manpower from one [machine] to another....
I have **overlooked** that here...on this particular project here....
And probably **wrongly so...I shouldn't have overlooked that**...which means we have to go back and redo that... [referring to setup violation]

The second type of difficulty is voiced several times.

Protocol E1, Task 2 (Difficult Postponement)

[18:18] Here's where I would make some alterations to run length...we don't need [part number] 25-cl at all...this is where I would begin maneuvering in this way [subject is referring to an adjustment based on priority]
[25:31] I almost need to go back to the idea of running these parts together [combine 29's and 24/25-25 on machine 2]...and that means altering the run length of the 29's to match that of 24/25-25...and I don't know how you could accommodate that with these rules.

Another tactic that is invoked by, but not available to, the experts in the experimental setting is "going to the call list," which is a listing of jobs sorted by inventory levels and compatibility. This allows E1 to add jobs to the schedule from a priority list which groups compatible or similar jobs together. In the experimental task the subject was given parts and their priorities, but not a call list.

Protocol E1, Task 2 (Difficult Postponement)

[22:14] What I'm doing now is checking what the possibilities are of not running 24, and what **I would normally do** is look for something else in this bend category that might be more feasible to run now, that doesn't have these additives.
[22:37] Fourteen seems to be an unusual bend category...we've got some coming with brackets and ceramics.
[22:55] So, we can't do that in that week...**that's when I would go to my call list**...I won't get into that.

The absence of valid operators to implement the strategy in the context of the experimental task provokes generation of non-valid states that the subject knows are not allowable. Worse, it induces E1 to ignore constraints that would make the solution infeasible in the experimental task setting. This results in time spent searching through invalid states (performance degradation in terms of total time) and constraint violations (performance degradation in terms of quality of end product).

6.2 Misinterpretation of Problem Features

The second source of degradation was from how jobs were represented on machines in the task materials. Changes in this aspect of the experimental task had strong effects on performance and behavior. For example, a job type with certain machine restrictions was represented as two split jobs (as it is the case in the real task) indicated by two separate cards (an artifact of the experiment). However, split jobs in the real-world version of the task are represented differently than was done in this task setting -- the information was equivalent in both representations, but presented differently. Unfortunately, E1 misinterprets the two separate cards as two different parts, and had many difficulties dealing with the split representation.

Protocol E1, Task 2 (Difficult Postponement)

[11:24] I'm evaluating the button and/or ceramic situation....**Oh! not so good.**
[11:35] No...**wait, this is all one part!** I really have trouble with that! I picture these as two different parts, count as two ceramics, but you can't do that.
[20:47] I have difficulties doing that [splitting up the 29's], because it would mean transferring [the same part] from one [machine] to another, and that's one thing we do avoid.

On task 4, once again the protocol of E1 shows that he has misinterpreted the problem features. In this case, he misinterpreted a single part as a combination part (two different parts run together).

Protocol E1, Task 4 (Easy Postponement)

[11:02] I was looking at this as a combination type, and it really isn't...just these are...these aren't two parts here, they're the same.

6.3 Phantom Constraints

The third source of degradation was the consideration of constraints from the real task that do not exist in the experimental task, and the ignoring of constraints from the experimental task that are not in the real task. Some constraints were eliminated in order to simplify the task for both novices and experts. For example, a constraint of when a job type may be run was relaxed in order to simplify the task of satisfying physical constraints in the scheduling of jobs on the machines. But this was obviously not the case for E1, who found it very difficult to ignore these constraints.

Protocol E1, Task 2 (Difficult Postponement)

[16:42] [subject checking priority list]...sunshade [25ss], stockout, oh! We are stockout, we have to run that, the clear [25cl] we don't need. **Those are the types of things I have to push aside**, and it's not easy!
[19:42] Another alternative I would consider...I would definitely just not run this [25cl], **it's what's giving problems.**

Another example of E1 carrying the constraints of the real task into the experimental task can be seen in this protocol excerpt from task 4.

Protocol E1, Task 4 (Easy Postponement)

[0:49] I'm just recognizing the fact that we probably don't even need 9 weeks supply, **with our constraints.**

The last two sources of degradation, misinterpretation of problem features and phantom constraints, are the result of small changes to features of the task such as the way of representing a job type or relaxing a type of constraint. Because these changes are small, the task remains similar enough to the original task so that the expert is "compelled" to invoke and apply previously acquired, but inappropriate, knowledge. In this manner, the expert is seduced into trying to solve the problem with an inappropriately configured problem space where, for example, an assumed constraint does not exist or where objects are represented differently than in the original context.

7. CONCLUSION

We propose that fragility of expertise in a complex, real-world task is characterized in terms of problem spaces and operators. In particular, we have shown that although the degradations of expertise in terms of performance do indeed occur (and are actually quite easy to induce), the notion of fragility as a generic comment on the relevance of knowledge is not warranted. By this we mean that the fragility of expertise is best viewed as "fragile" only in the context of performance and not in terms of behavior. By distinguishing behavior from performance, we discovered that interactions between task and level of performance previously used as evidence to describe expert fragility are actually missing a very important point -- that even if performance is degraded, expert behavior is persistent. This is because the failure of expert knowledge is very localized and particular. Expert knowledge failure occurs because KEY and SPECIFIC components fail, not because of a catastrophic invalidation of an entire knowledge base. Furthermore, this failure occurs in particular (and describable) ways -- inappropriate problem space characterizations and problems with operator implementations or availability.

Designers of expert systems should be aware of the implications of the fragility of expertise. These implications touch on two main issues: knowledge acquisition and representation. The design of an expert support system includes a phase of knowledge acquisition, during which domain specific knowledge will be accumulated by the designer. This acquisition phase will often include the observation of an expert performing the task(s) of interest, but it may be that the expert's normal environment is not suitable for such observations (the expert may be distracted by frequent requests for his/her expertise, or the physical environment itself may be distracting). In this case of an unsuitable environment, the designer may fashion a simulated task for the expert to perform in an environment more suited to precise observation. The designer must be extremely careful in the design of this simulated task for, as demonstrated by this study, even seemingly small changes from the real task to the simulated task may bring about large performance degradations, and the knowledge acquired though the observation of such a degraded performance will not be of much use.

Once the expert system is actually being designed, the issue of knowledge representation comes into play. A representation scheme is usually designed with the intent of simplifying the task for the user, and so the designer may choose to leave seemingly unimportant features of the task environment out of the representation. However, such simplifications may actually hinder the performance of an expert, as shown by this study. Not only might such simplifications cause conflict with the knowledge of the expert, but the expert may waste cognitive resources by having to keep track of those features that are not in the support system's representation.

We conclude by noting that the phenomena we have presented in this paper are being further investigated. Specifically, in this paper we proposed that expert fragility could be explained when one cast the task in terms of the problem space hypothesis set forth by Newell and Simon (1972). A second set of experiments are being conducted based on this hypothesis; however, these experiments involve an artificial intelligence model of the scheduling process which incorporates learning from experience (Hsu, Prietula and Steier 1989; Prietula et al. 1989). Furthermore, recent (and related) efforts at incorporating a problem space approach for facilitating expert system development reflect an important step in characterizing how humans reason about tasks and how knowledge

engineers may formally express that knowledge (Yost and Newell 1989). With both human and modeling data in hand, we hope to further gain insight into the nature of expertise and how expertise can be represented in computational form.

8. ACKNOWLEDGEMENTS

We gratefully acknowledge the helpful comments given by Bonnie John of Carnegie Mellon University.

9. REFERENCES

Adelson, B. "When Novices Surpass Experts: The Difficulty of a Task May Increase With Expertise." *Journal of Experimental Psychology: Learning, Memory, and Cognition*, Volume 10, Number 3, 1984, pp. 483-495.

Brown, A., and Campione, J. "Three Faces of Transfer: Implications for Early Competence, Individual Differences, and Instruction." In M. Lamb, A. Brown and B. Rogoff, Editors, *Advances in Developmental Psychology*, Hillsdale, New Jersey: Lawrence Erlbaum, 1984.

Chase, W., and Simon, H. "Perception in Chess." *Cognitive Psychology*, Number 4, 1973.

Chi, M.; Feltovich, P.; and Glaser, R. "Categorization and Representation of Physics Problems by Experts and Novices." *Cognitive Science*, Number 5, 1981, pp. 121-152.

Douglas, S. A., and Moran, T. P. "Learning Text Editor Semantics by Analogy." In *Proceedings of the Computer Human Interaction Conference-CHI 83*, 1983, pp. 207-211.

Ericsson, K. A., and Simon, H. A. "Verbal Reports as Data." *Psychological Review*, Number 87, 1980, pp. 215-251.

Hsu, W. L.; Prietula, M.; and Steier, D. "Merl-SOAR: Scheduling Within a General Architecture for Intelligence." To appear in *Proceedings of the 3rd International Conference on Expert Systems and the Leading Edge in Production and Operations Management*, Hilton Head Island SC, May 21-24, 1989.

Johnson, P.; Duran, A.; Hassebrock, F.; Moller, J.; Prietula, M.; Feltovich, P.; and Swanson, D. "Expertise and Error in Diagnostic Reasoning." *Cognitive Science*, Number 5, 1981, pp. 235-283.

Larkin, J.; McDermott, J.; Simon, D.; and Simon, H. "Expert and Novice Performance in Solving Physics Problems." *Science*, Number 208, 1980, pp. 1335-1342.

McKeithen, K. B.; Reitman, J. S.; Rueter, H. H.; and Hirtle, S. C. "Knowledge Organization and Skill Differences in Computer Programmers." *Cognitive Psychology*, Volume 13, Number 3, 1981, pp. 307-325.

Newell, A. "Heuristic Programming: Ill-structured Problems." In J. Aronofsky, Editor, *Progress in Operations Research: Volume 3*, New York: Wiley, 1969.

Newell, A. "Physical Symbol Systems." *Cognitive Science*, Number 4, 1980a, pp. 135-183.

Newell, A. "Reasoning, Problem Solving and Decision Processes: The Problem Space as a Fundamental Category." In R. Nickerson, Editor, *Attention and Performance VIII*, Hillsdale, New Jersey: Lawrence Erlbaum, 1980b.

Newell, A., and Simon, H. *Human Problem Solving*, Englewood Cliffs, New Jersey: Prentice-Hall, 1972.

Prietula, M.; Hsu, W. L.; Newell, A.; and Steier, D. "Sharing Scheduling Knowledge Between Intelligent Agents." *AI and Manufacturing Workshop, International Joint Conference on Artificial Intelligence*, Detroit, Michigan, 1989.

Prietula, M., and March, S. *Form and Substance in Physical Database Design: An Empirical Study of Problem Solving*, Manuscript Submitted for Publication, 1989.

Reed, S.; Ernst, G.; and Banerji, R. "The Role of Analogy in Transfer Between Similar Problem States." *Cognitive Psychology*, Number 6, 1974, pp. 436-450.

Simon, H. "Cognitive Science: The Newest Science of the Artificial." *Cognitive Science*, Number 4, 1980, pp. 33-46.

Simon, H. "Information Processing Theory of Human Problem Solving." In W. Chase, Editor, *Handbook of Learning and Cognitive Processes, Volume 5: Human Information Processing*, Hillsdale, New Jersey: Lawrence Erlbaum, 1978.

Simon, H. *The Sciences of the Artificial*. Cambridge, Massachusetts: MIT Press, 1981.

Simon, D., and Simon, H. "Individual Differences in Solving Physics Problems." In R. Siegler, Editor, *Children's Thinking: What Develops?*, Hillsdale, New Jersey: Lawrence Erlbaum, 1978.

Yost, G. R., and Newell, A. "A Problem Space Approach to Expert System Specification." To appear in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, August, 1989.

10. ENDNOTES

1. The phrase "information processing psychology" basically refers to a general theory of human problem solving put forth by Newell and Simon (1972) which had as its focus the principle that human cognition could be viewed, modeled, and explained in terms of an information processing system. A relatively recent phrase, "cognitive science," refers to a multidisciplinary view of studying cognition (human or machine) which subsumes information processing psychology (cf. Simon 1980). The distinctions, however, are becoming increasingly blurred.

2. *Reactive* scheduling (as opposed to "generative" scheduling) requires changes to be made to an existing schedule in response to unanticipated events which render the current schedule inadequate. Typically, reactive scheduling problems have an added difficulty in that it is important to minimize the disruption (i.e., change to) the existing schedule.