On the Roles of Search and Learning in Time-Limited Decision Making

Susan L. Epstein
Department of Computer Science
Hunter College and The Graduate School of The City University of New York
New York, NY 10021
epstein@roz.hunter.cuny.edu

Abstract

Reported properties of human decision-making under time pressure are used to refine a hybrid, hierarchical reasoner. The resultant system is used to explore the relationships among reactivity, heuristic reasoning, situation-based behavior, search, and learning. The program first has the opportunity to react correctly. If no ready reaction is computed, the reasoner activates a set of time-limited search procedures. If any one of them succeeds, it produces a sequence of actions to be executed. If they fail to produce a response, the reasoner resorts to collaboration among a set of heuristic rationales. A time-limited maze-exploration task is posed where traditional AI techniques fail, but this hybrid reasoner succeeds. In a series of experiments, the hybrid is shown to be both effective and efficient. The data also show how correct reaction, time-limited search with reactive trigger, heuristic reasoning, and learning each play an important role in problem solving. Reactivity is demonstrably enhanced by brief, situation-based, intelligent searches to generate solution fragments.

1. Introduction

When confronted with a difficult problem and a limited amount of time to decide upon an action, people employ a variety of devices to make what they hope will be expert decisions. Some of this behavior is automatic; perceptions about the current state of the world may trigger a response without conscious reasoning. AI researchers model such automaticity with *reactive* systems. Other portions of this behavior are *heuristic*; an approximately correct decision rule is selected and applied. AI researchers model such "rules of thumb" with rule-based systems. There is, however, another important mechanism people use. *Situation-based behavior* is the serial testing through search of known, triggered techniques for problem solving in a domain. This paper describes a cognitive model that integrates situation-based behavior with reactivity, heuristic reasoning, and learning. The contributions of this work are the model and empirical evidence from it that situation-based behavior is an effective method for decision-making under time pressure.

Situation-based behavior is based upon psychologists' reports about human experts in resource-limited situations (Klein & Calderwood, 1991). For example, an emergency rescue team is called to the scene of an attempted suicide, where a person dangles from a sign after jumping from a highway overpass. Time is limited and the person is semiconscious. During debriefing after a successful rescue, the commander of the team describes how they immediately secured the semiconscious woman's arms and legs, but then needed to lift her to safety. He retrieved, instantiated, and mentally tested four devices that could hold her while the team lifted, one device at a time. When a device failed in his mental simulation, he ran the next. When the fourth scenario ran several times in simulation without an apparent flaw, he began to execute it in the real world. Klein and Calderwood describe the predominance of this situation-based behavior in 32 such incidents, and cite additional evidence from studies of other decision-makers under time pressure. Its key features, for the purposes of this discussion, are that a situation triggers a set of procedural responses, not solutions, and that those responses are not tested in parallel.

The purpose of this paper is not to argue that situation-based behavior is the only way to reason and search, but to explore its role with respect to learning and reactivity and heuristic reasoning under time limitations. The next section describes the integration of situation-based behavior into a problem solving and learning model called FORR. Subsequent sections detail the problem domain, describe an implementation for path finding, discuss the experimental results, and relate situation-based behavior to other work.

2. FORR: the Model

The problem solvers Klein and Calderwood studied did not have the leisure to research similar situations or to explore many alternatives. They had to decide quickly. Reactive systems are intended to sense the world around them and respond with a quick computation (Brooks, 1991; Maes & Brooks, 1990). They iterate a "sense-compute-execute" loop where the sensing is predetermined and the heuristic computation is either hardwired or extremely rapid. In most complex dynamic problems, however, a simulation of intelligence is strengthened by learning. In the spirit of reactivity, such learning should be quick to do and easy to apply in subsequent loop iterations. The model described in this section supports the development of such a reactive reasoner.

FORR (FOr the Right Reasons) models the transition from general expertise to specific expertise (Epstein, 1994). A FORR-based system begins with a *domain* of related problem classes, such as board games or mazes, and some domain-specific but problem-class-independent knowledge, such as "do not set the other contestant up for a win" or "avoid dead-ends." With problem solving experience, such as contests played or trips from one maze location to another, a FORR-based program acquires *useful knowledge*, problem-class-specific data that is potentially useful and probably correct. Useful knowledge, such as good game openings or shortcuts in a particular maze from one vicinity to the next, should enhance the performance of a FORR-based system.

FORR integrates reactivity, situation-based behavior, and heuristic reasoning in the three-tiered hierarchical model

shown in Figure 1. Tier 1 is reactive and correct, tier 1.5 is situation-based search, and tier 2 is reactive and heuristic. An *Advisor* epitomizes a domain-specific but problem-class-independent, decision-making rationale, such as "minimize the other contestant's material" or "get closer to your destination." Each Advisor is a "right reason" for decision-making in the domain, implemented as a time-limited procedure. Input to each Advisor is the current state of the world, the current permissible actions from that state, and any learned useful knowledge about the problem class under consideration. Each Advisor outputs any number of *comments* that support or discourage permissible actions. A comment lists the Advisor's name, the action commented upon, and a *strength*, an integer from 0 to 10 that measures the intensity and direction of the Advisor's opinion.

FORR addresses a problem as a sequence of decisions to be made. At decision-making time, a FORR-based system senses the current state of the world and reacts with a rapidly-computed decision. The calculation for that decision begins in the first tier, where Advisors are consulted in a predetermined, fixed order. They may have the authority to make a decision unilaterally or to eliminate a legal action from any further consideration. First-tier Advisors are reactive, consulted in sequence, and reference only correct useful knowledge. They "sense" the current state of the world and what they know about the problem class; if they make a decision, it is fast and correct. The commander had a first-tier Advisor which insisted that the victim's limbs be secured. A good first-tier Advisor for game playing is "if you see a winning move, take it;" a good one for maze-traversing is "if you see the goal, go to it." Only when the first tier of a FORR-based system fails to make a decision does control default to the next tier.

Second-tier Advisors, in contrast, are not necessarily independent, or even correct in the full context of the state space. Each of them epitomizes a heuristic, specialized view of reality that is a reasonable argument for or against one or more actions. Second-tier Advisors are reactive too, but far less trustworthy, because neither their reasoning process nor the useful knowledge on which they rely is guaranteed correct. All the second-tier Advisors have an opportunity to comment before any decision is made. The decision they compute is the action with the highest total strength; this represents a consensus of their opinions. A good second-tier Advisor for game playing is "maximize the number of your pieces on the board and minimize those of the other contestant;" a good one for maze-traversing is "move in the direction of the goal." There is evidence in the literature that people approach complex tasks as if they had such Advisors (Biswas, Goldman, Fisher, Bhuva, & Glewwe, 1995; Ratterman & Epstein, 1995), but for the rescue situation, second-tier Advisors may be too slow and too risky.

Situation-based behavior has recently been incorporated into FORR with tier 1.5. A tier-1.5 Advisor temporarily digresses from the "sense-compute-execute" loop when it recognizes a situation in which its knowledge may have substantial impact. Each tier-1.5 Advisor has a reactive trigger to recognize that the current situation may benefit from its solution method, just as the need to hoist was a trigger for each of the rescue team's holding devices. Each also has a

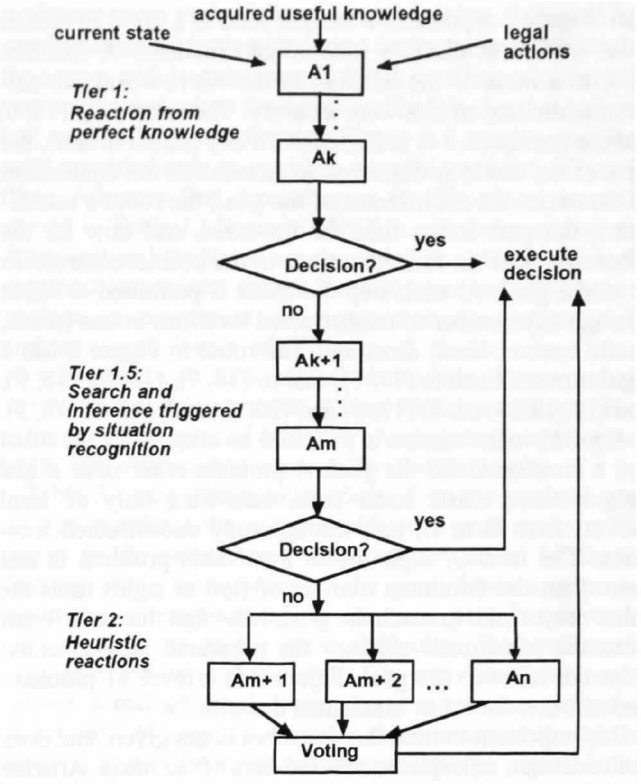

Figure 1: How FORR makes decisions.

knowledge-intensive, highly-constrained search procedure to construct a *solution fragment*, a sequence of decisions rather than a single reactive one. If the first tier has failed to make a decision, each of the prioritized Advisors of tier 1.5 has the opportunity in turn to trigger, and each triggered Advisor is ceded control until one of them produces a solution fragment. Tier-1.5 Advisors execute decisions in the actual problem space as they attempt to construct a solution fragment. Regardless of its outcome, the first returned solution fragment is incorporated into the solution under construction, and control is returned to tier 1. If no tier-1.5 Advisor produces a sequence of recommended steps, the second tier will make the decision. The effectiveness of situation-based Advisors and their role in reasoning is best demonstrated with an example.

3. Ariadne: an Implementation

Ariadne is a FORR-based system for simulated robot path-finding. (Ariadne, daughter of King Minos, helped Theseus find his way through the labyrinth.) Ariadne models learning the way around a complex geographic area through a series of trips there. A problem class for Ariadne is a particular maze, and learning occurs as a result of repeated path finding in the same maze. Ariadne's task is to move a robot from some initial location to the stationary goal in a sequence of legal moves. The robot, however, is severely restricted. It has no explicit map and it is not permitted to construct one. It senses where it is and where it can go in one step, decides which step to take, and "leaps" there without collecting data on the way.

A *state* in Ariadne is a maze containing the robot and the

goal. Figure 2 represents a sample state in a 20 × 20 rectangular grid with discrete internal obstructions. A *location* (r, c) in a maze is the position in the *r*th row and *c*th column, addressed as if it were an array. The robot is at (18, 6) and the goal at (5, 14) in Figure 2. At any instant in time, the state of the world is described to Ariadne as the dimensions of the maze, the coordinates of the goal, the robot's coordinates, the path it has thus far traversed, and how far the robot can "see" in four directions to the nearest obstruction or to the goal. At each step the robot is permitted to move through any number of unobstructed locations in one (north, south, east, or west) direction. The robot in Figure 2 has 8 legal moves: north to (17, 6), east to (18, 7), (18, 8), (18, 9), and (18, 10), south to (19, 6) and (20, 6), and west to (18, 5).

A *problem* in Ariadne is an initial location R for the robot and a location G for the goal. A problem is *solvable* if and only if there exists some path, consisting only of legal moves, from R to G, i.e., through only unobstructed locations. The *level of difficulty* of a solvable problem is one more than the minimum number of (left or right) turns the robot must make to reach the goal. Note that this is different from the *Manhattan distance* (as measured in grid units) from the robot to the goal. Figure 2 is a level 11 problem; one solution for it has Manhattan distance 29.

It is important to note that the robot is not given, and does not construct, an explicit, detailed map of the maze. Ariadne does, however, learn descriptive abstractions about the maze as useful knowledge: gates, dead-ends, and chambers. A *gate* is a location that offers a transition from one quadrant of the maze to another, for example, (11, 3) is a gate between quadrants 3 and 2 in Figure 2. Of course, a gate may not always be helpful; (11, 3) offers access to little of quadrant 2. After each decision, Ariadne tests whether its last move has changed its quadrant, that is, if it has moved through a gate. If so, the robot's current location is learned as a gate between the current quadrant and the previous one. A *corridor* is a passageway of width one that either leads nowhere (a *dead-end*) or is a *hallway*. In Figure 2 {(14, 1), (15, 1), (16, 1)} is a dead-end and {(5, 15), (5, 16), (6, 16), (6, 17)} is a hallway that zigzags. A corridor is learned as a pair of endpoints when, from the current state, the robot has only one or two moves. Corridors are enlarged and merged together as necessary. A *chamber* is an irregularly shaped space with an access point and an approximate *extent,* the furthest in each direction one can go in the chamber. This is a compact, heuristic description that at worst overstates the chamber by a bounding rectangle. Figure 2's robot is in a chamber with access point (16, 5) and extent 16 north, 10 east, 20 south, and 4 west. The *access point* of a chamber is a location within the chamber that affords a view outside it. For example, from (16, 5) the robot can see east beyond its extent to (16, 3). All locations reachable from the robot really constitute one large chamber, but the chambers that Ariadne learns are more limited and room-like. When Ariadne is not making good progress and the decision cycle reaches tier 1.5, an Advisor may call a search procedure that scans vertically and horizontally from the robot's current position to estimate the extent of the current chamber. A chamber is represented as an extent-access-point pair and stored on a list. A new chamber may subsume an old one, in which case it replaces it on the list. Otherwise, chambers are not merged, and they may overlap or have more than one access point.

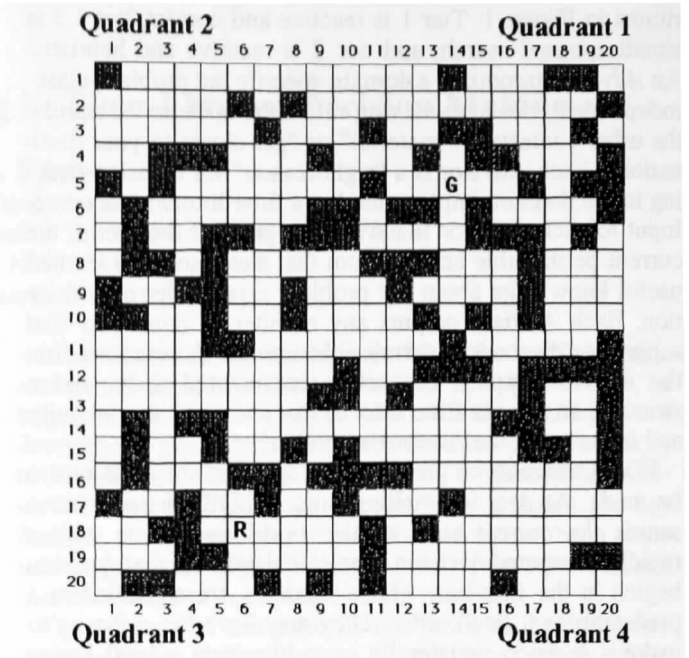

Figure 2: An Ariadne problem. The robot must move to the goal in unidirectional steps through unobstructed locations.

Ariadne has 17 Advisors, listed in Table 1, with tiers 1 and 1.5 in order of their relative priority. Descriptions of the triggers for the tier-1.5 Advisors are italicized. For example, Roundabout triggers when the robot is in the same row or column as the goal but it cannot see it because of an obstruction. Roundabout attempts to shift over and then go around the wall between it and the goal. If, for example, the robot of Figure 2 were at (5, 18), Roundabout would take it to (6, 18), (6, 17), and (6, 16) before stopping at (5, 16) where the goal is in sight. Note that Roundabout, like any tier-1.5 Advisor, is time-limited and heuristic. It may fail, or it may only get closer to the goal than it had been, without actually bringing the goal in sight.

Ariadne's tier-2 Advisors embody path-finding commonsense and do no forward search at all in the problem space. Chamberlain, for example, encourages moves to the entry point of a dead-end or a chamber whose extent indicates that the goal might lie within, and discourages moves to any other entry points. (Although Chamberlain's comments are based upon heuristic extents, it never permanently prevents a solution, because the other Advisors may eventually override it with their own comments.) Note that Chamberlain, unlike Outta Here, is not permitted to search. The simple ideas behind the tier-2 Advisors support rapid computation. FORR signals (but continues to calculate) if any Advisor runs out of time, and that has yet to happen with Ariadne.

Ariadne is implemented as a set of Common Lisp routines that run with FORR. To create a problem class, the user specifies the dimensions of the maze, the level of problem difficulty, and the percentage of internal obstruction. Further technical details are available in (Epstein, 1995).

Table 1: Ariadne's Advisors for path finding. Starred Advisors reference useful knowledge.

Name	Description
Tier 1	
No Way*	Do not enter a goal-free dead-end.
Victory	Move to the goal if it is in sight.
Tier 1.5	
Roundabout	Move around the wall *if already in the right row or column.*
Outta Here*	Exit a *goal-free* dead-end or chamber or *a small geographical area.*
Probe*	Exit a chamber *if recently locally constrained.*
Super-Quadro*	Exit a quadrant *if recently locally constrained.*
Wander	Take very large steps L-shaped steps *if recently locally constrained.*
Tier 2	
Been There	Avoid return to a previous location.
Chamberlain*	Avoid or encourage entrances to dead-ends and chambers based upon their extent.
Done That	Avoid repetition of a move in the same direction as one already taken from a previous location.
Giant Step	Make the longest possible move in some direction.
Goal Row	Move to the same row as the goal.
Goal Column	Move to the same column as the goal.
Mr. Rogers	Move as close to the goal as possible.
Opening*	Begin the way a previous successful path did.
Plod	Move one location in some direction.
Quadro*	Move from one quadrant to another through known gates.

4. Experimental Results

The performance of six reasoning agents was tested: the full version of Ariadne and five ablated versions that measure the contribution of Ariadne's various components. The *Random agent* selects random legal moves; this is equivalent to blind search. The *Reactive agent* learns, but uses only the Advisors in tier 1; it simulates correct reactive response. The *Reactive+ agent* learns, but uses only the Advisors in tiers 1 and 1.5; it simulates reactive decision making with situation-based behavior but without heuristic reasoning. The *Reactive-Heuristic agent* learns, but uses only the Advisors in tiers 1 and 2; it simulates reactive decision making without situation-based behavior. *No-Learning* includes only the Advisors in any tier that do not consult learned useful knowledge (those starred in Table 1). The *FORR agent* uses all the Advisors in Table 1; this simulates reactive decision making and learning with situation-based behavior. Agents were eliminated from testing after poor performance. During early trials the Random agent, solved only 12% of 100 level 6 problems; it therefore served merely as a benchmark.

A *run* for a fixed, randomly-generated maze is 10 learning problems given to the full version of Ariadne, followed by 5 newly-generated testing problems in the same maze given to the agents with learning turned off. A problem of either kind was terminated when the agent reached the goal or when it had made 100 passes through Figure 1. Learning problems were provided only to establish a useful knowledge base for those Advisors that depend upon it. (Those Advisors are starred in Table 1.) Because FORR is non-deterministic and the mazes and problems are generated at random within the specified constraints, results from 10 runs were averaged to produce an *experiment.* An experiment was performed for problems with levels of difficulty 6, 8, 10, and 12 in a 20×20 maze with 30% internal obstruction. These parameters were selected to provide at least 1000 possible problems at the specified level of difficulty. Any agent that performed badly was omitted from more difficult experiments.

Table 2 reports the results. "Solved" is the percentage of the test problems the agent could solve with at most 100 moves in the same maze. "Path length" is the Manhattan distance along the solution to the goal. Since a step may move through more than one location, path length varies among problems of the same difficulty. "Moves" is the number of moves in the solution. The number of distinct locations actually visited during those moves is reported as "locations." "Triggers" measures the reliance of the system on tier 1.5; it is the number of passes through Figure 1 during which any situation-based Advisor executed. Path length, moves, and locations are computed only over solved problems. (This makes the ablated agents look somewhat better than they actually are.) "BFS%" is the percentage of the space reachable from the robot's initial position that breadth-first search would have visited on the same test problems. Time pressure can be applied to a solution two ways: either as the path length (since computation time would be much faster than travel time), or as the number of passes through Figure 1 (based purely on computation time).

As the problems become more difficult, the ability of the ablated agents to solve the problems becomes markedly inferior, and the situation-based Advisors trigger more often. The Reactive-Heuristic agent, FORR's original formulation, draws the robot to fewer locations and constructs shorter paths than Reactive+ on the simpler problems, but solves fewer difficult ones. Although situation-based Advisors make some contribution when combined with tier 1, Reactive+ is clearly inadequate on the more difficult problems. The situation-based Advisors trigger more often with Reactive+ than with the full FORR agent because most of them recognize repetitive action, and Reactive+ frequently behaves repetitively.

The full FORR agent is clearly more powerful than the ablated ones. FORR with tier-1.5 offers a measure of reliability and achievement the other versions lack. The number of successes by the full FORR agent represents a statistically significant improvement over the others. Although this work was predicated on the acceptability of suboptimal solutions, the successful paths of the ablated agents are extremely long. With all of FORR's tiers in place, Ariadne gets the robot to the goal more often, more quickly, and considers fewer alternatives along the way.

Table 2: Average testing performance of agents after learning in 10 randomly generated 20 × 20 mazes. Search terminated upon solution or after 100 decisions.

Level	Agent	Solved	Path Length	Moves	Locations	Triggers	BFS %
6	Reactive	24%	156.2	52.0	29.8	—	66.1%
	Reactive+	96%	61.2	28.0	22.0	13.3	
	Reactive-Heuristic	90%	48.2	23.5	16.1	—	
	FORR	98%	29.7	19.2	15.8	5.5	
8	Reactive+	86%	79.7	37.4	29.1	17.7	87.4%
	Reactive-Heuristic	88%	93.3	37.5	23.7	—	
	FORR	96%	45.5	28.5	24.3	9.9	
10	Reactive+	80%	105.3	50.4	37.8	19.7	95.2%
	Reactive-Heuristic	66%	122.0	54.3	33.0	—	
	FORR	86%	60.6	38.3	28.5	14.8	
12	Reactive+	64%	118.0	53.0	41.2	29.9	96.2%
	FORR	80%	69.4	41.8	31.7	25.7	

5. Search and Learning

This domain is not amenable to traditional AI search techniques. Depth-first search requires elaborate backtracking and loop prevention to calculate any solution; very few, if any, of the test problems would be solvable in 100 steps this way. Breadth-first search, while it will always solve the problem, does so at the cost of visiting a high proportion of the nodes ever accessible to the robot from its starting location in the search space. AI searches are often steered by an evaluation function toward the "most promising" locations to avoid such difficulties. The robot's knowledge is so limited, however, that an evaluation function would have all the shortcomings of the Reactive-Heuristic agent. For example, closer to the goal is not necessarily better; there may be a very long wall there. Means-ends analysis, another standard AI technique, is not possible because the robot knows little, if anything at all, about the immediate vicinity of the goal. For a very large maze, then, explicit search would be extremely inefficient, perhaps intractable.

There is a complex relationship among the tiers. Tier 1.5 requires both tier 1's commonsense and tier 2's heuristic knowledge to be effective. Tier 2 tries to avoid search and effectively sets up the situation-based Advisors in tier 1.5 so that they can trigger. For example, Goal Row and Goal Column push the robot into a situation where Roundabout can trigger. In turn, the situation-based Advisors of tier 1.5 set up the heuristic reasoners in tier 2. For example, Wander puts the robot where all the tier-2 Advisors are more likely to make new, constructive comments. When Ariadne bogs down, the triggers of the tier 1.5 Advisors behave like a search party; they expend resources to improve the program's ability to make progress. The useful knowledge acquired this way is not overwhelming. Ariadne averaged only 30.5 corridors and 10.8 chambers in its level 12 problems.

The initial impulse behind reactive programming was to avoid search. When one augments the reactive Advisors of tier 1 and tier 2 with tier 1.5, the system is kept within the search minimization philosophy two ways. First, FORR only allocates each Advisor, in any tier, a limited amount of computing time. Therefore solution fragments that take too long to construct are not considered. Second, tier-1.5 Advisors have hand-coded routines intended to address their particular subgoals. These routines generate and test solution fragments, just the way the commander did, but the proposed partial solutions must be highly constrained, just as the commander's are. This constraint saves the tier-1.5 Advisor from a combinatoric explosion. Although their search can be quite deep, they are effective because they are severely curtailed by knowledge.

An important difference between FORR with tier 1.5 and the commander is the fact that he ran his successful simulation four times before he implemented it. Comparing moves and locations in Table 2, it is clear that Ariadne could shorten its path lengths by as much as 25% if it removed the loops. With respect to timing, however, the robot visited those locations, so the entire path is still the cost.

The role of learning in this domain becomes apparent only in the most difficult problems. In another series of experiments we tested the full FORR agent against No-Learning. By level 10, No-Learning solved 20% fewer problems and triggered tier 1.5 three times as often. There were mazes (runs) where No-Learning could solve all five level-10 problems (albeit in slightly longer paths) without learned useful knowledge, but in two mazes it could solve only two. The paths No-Learning finds in hard problems look like a flight of steps. No-Learning fails on problems where the goal is hidden behind or the robot begins behind a variety of deceptive barriers. Even when No-Learning could solve a hard problem, the solutions with learning were shorter, less repetitive, and required fewer decision cycles.

Ariadne has already performed well on preliminary tests in 30 × 30 mazes and continues to improve as we refine its Advisors and its learning algorithms. There is every reason to believe that Ariadne will continue to scale up, i.e., perform well in much larger and more tortuous mazes than these. Hoyle, a FORR-based game-learning program, progressed to much larger spaces after the addition of only a few tier-2 Advisors (Epstein, 1994).

6. Related Work

Situation-based behavior is not case-based reasoning (*CBR*), although they have much in common. In CBR, experiences are indexed and stored. Later, when a problem arises, one or more potentially relevant cases that "remind" the system of the current one are retrieved, and an attempt is made to modify their solutions to solve the current problem (Kolodner, 1993). Although situation-based behavior is triggered by an abstraction of the current state that could have been used as an index for CBR, situation-based behavior does not retrieve specific solutions to be modified, only procedures intended to generate solution fragments. Situation-based behavior and CBR both constrain solution generation, but CBR does it by searching from old solutions, while situation-based behavior does it by the knowledge inherent in its procedures. Klein and Calderwood emphasize that the human experts they study do not perceive their problem solving as reminding. This is not a claim that CBR has no parallel in people, only that it is less likely to be used under time pressure.

Situation-based behavior is not planning either. A plan is a set of actions intended to reach a specific goal. The commander tested holding devices by incorporating them into plans and mentally executing those plans until one promised success. The situation-based Advisors of tier 1.5 are not planners because they actually execute their behavior, even if they do not eventually recommend it. For example, Wander can investigate as many as eight L's (by moving one longest step in each direction and then testing for possible second steps) before it chooses one to execute. Rather than planners, situation-based Advisors are procedures that reactively seize control of a FORR-based program's resources for a fixed period of time. When that time elapses the situation-based Advisor either returns control to tier 2 or returns a sequence of actions whose execution it requires. Tier 2 constitutes a reactive decision maker, much like Pengi (Agre & Chapman, 1990). The principal difference is that Pengi's problem is living in its world; it is not held to an explicit decision standard like Ariadne's "solve in 100 decision steps."

Nor is situation-based behavior a macro-operator. A macro-operator is a generalization across the variables entailed in a successful procedure, whereas a situation-based Advisor is a procedural generalization over several kinds of behavior appropriate to a situation. Situation-based behavior is a resource-grabbing heuristic digression intended to produce a solution fragment.

Situation-based behavior does shed some light on the debate about representation and reactivity (Hayes, Ford, & Agnew, 1994). For Ariadne conceptual knowledge includes situation-based triggers like "the last 30% of the moves have been in no more than 5% of the locations in the maze" and "a wall lies between the aligned robot and the goal." This work demonstrates that, at least in this domain, the representation of conceptual knowledge is an essential component in a reactive learner.

7. Conclusions

Ariadne succeeds on time-limited decision problems where traditional AI techniques fail. Ablation indicates that no single component of the hybrid reasoner is responsible for its success; there is a synergy among them. The program is robust; it can learn in any maze without "tuning."

The thesis of this work is that severely constrained search, particularly when enhanced by learning, can play an important role in the performance of an otherwise reactive and heuristic system. Situation-based behavior is modeled on people who produce suboptimal solutions under time constraints. A reactive system goes directly from perception to an associated action, without any opportunity to reason about the state. With tier 1.5, FORR, like Klein and Calderwood's subjects, perceives and then reasons about the current state of the world before it elicits an associated action, but still maintains some of the advantages of reactivity. Ariadne's success at maze search is a clear indication that resource allocation to highly-restricted, intelligent search is an important facet in the simulation of efficient, effective learning and decision making under resource limitations.

Acknowledgments

Jack Gelfand originally formulated a somewhat different Ariadne problem, and continued to ask constructive questions as this research evolved. David Sullivan's preliminary work on Tonka convinced us that a FORR-based version of Ariadne could be a success. Rich Korf, Jim Hendler, and Rich Sutton offered helpful comments.

References

Agre, P.E., & Chapman, D. (1990). What are Plans for? *Robotics and Autonomous Systems, 6*, 17-34.

Biswas, G., Goldman, S., Fisher, D., Bhuva, B., & Glewwe, G. (1995). Assessing Design Activity in Complex CMOS Circuit Design. In P. Nichols, S. Chipman, & R. Brennan (Ed.), *Cognitively Diagnostic Assessment.* Hillsdale, NJ: Lawrence Erlbaum.

Brooks, R.A. (1991). Intelligence without Representation. *Artificial Intelligence*, *47*(1-3), 139-160.

Epstein, S.L. (1994). For the Right Reasons: The FORR Architecture for Learning in a Skill Domain. *Cognitive Science*, *18*(3), 479-511.

Epstein, S.L. (1995). On Heuristic Reasoning, Reactivity, and Search. *Proceedings of IJCAI-95* (in press). San Mateo: Morgan Kaufmann.

Hayes, P.J., Ford, K.M., & Agnew, N. (1994). On Babies and Bathwater: A Cautionary Tale. *AI Magazine*, *15*(4), 15-26.

Klein, G.A., & Calderwood, R. (1991). Decision Models: Some Lessons from the Field. *IEEE Transactions on Systems, Man, and Cybernetics*, *21*(5), 1018-1026.

Kolodner, J.L. (1993). Introduction to the Special Issue on Case-Based Reasoning. *Machine Learning*, *10*(3), 1-5.

Maes, P., & Brooks, R.A. (1990). Learning to Coordinate Behaviors. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 796-802). Palo Alto, CA: AAAI Press.

Ratterman, M.J., & Epstein, S.L. (1995). Skilled like a Person: A Comparison of Human and Computer Game Playing. *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society* (in press). Hillsdale, NJ: Lawrence Erlbaum Associates.