On Heuristic Reasoning, Reactivity, and Search

Susan L. Epstein
Department of Computer Science
Hunter College and The Graduate School of The City University of New York
695 Park Avenue
New York, NY 10021
U. S. A.

Abstract

This paper explores the relationships among reactivity, heuristic reasoning, and search. It describes a hybrid, hierarchical reasoner that first has the opportunity to react correctly. If no ready reaction is computed, the reasoner activates a set of reactive triggers for time-limited search procedures. If they too fail to produce a response, the reasoner resorts to collaboration among a set of heuristic rationales. In a series of experiments, this hybrid reasoner is shown to be effective and efficient. The data also show how each of the three processes (correct reactions, time-limited search with reactive trigger, and heuristic rationales) plays an important role in problem solving. Reactivity is demonstrably enhanced by brief, knowledge-based, intelligent searches to generate solution fragments.

1. Introduction

When confronted with an intractable search space, people employ a variety of devices to make what they hope will be expert decisions. Some of this behavior is automatic; certain perceptions of the world trigger action without conscious reasoning. AI researchers have modeled this automaticity with *reactive systems.* Other portions of this behavior are heuristic; limitedly rational reasoning principles are applied in some combination. AI researchers have modeled this pragmatic behavior with rule-based systems. There is, however, another important mechanism people use. *Situation-based behavior is* the serial testing of known, triggered techniques for problem solving in a domain. The integration of situation-based behavior with reactivity and heuristic reasoning is the subject of this paper. The contributions of this work are an architecture that integrates situation-based behavior with reactivity and heuristic reasoning, and empirical evidence that situation-based behavior is indeed an effective method when resources are limited.

Situation-based behavior is based upon psychologists' reports about human experts in resource-limited situations [Klein and 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 army commanders, business executives, juries in deliberation, judges setting bail, highway engineers, and nuclear power plant operators. 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 to explore the role of situation-based behavior with respect to reactivity and heuristic reasoning under time limitations.

2. FORR: the Architecture

The architecture described in this section supports the development of a reactive reasoner. The problem solvers whom 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 and Brooks, 1990]. They are meant to 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.

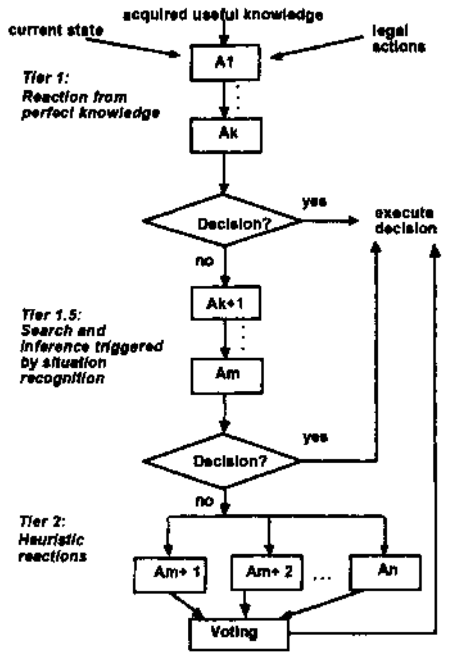
FORR (FOr the Right Reasons) is a problem-solving and learning architecture that models the transition from general expertise to specific expertise [Epstein, 1994a]. A FORR-based system begins with a *domain* of related problem classes, such as board games or mazes, and some domain-specific but problem-class-independent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

FORR's three-tiered hierarchical model of the reasoning process, is shown in Figure 1. An *Advisor* is 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," 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 current problem class. 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. Although there are no constraints on the nature of the comment-generating procedures themselves, a FORR-based system is intended to sense the current state of the world and react with a rapid computation, i.e., to avoid extensive search.

Tier-1 Advisors are consulted in a predetermined, fixed order. Each Advisor may have the authority to make a decision alone or to eliminate a legal action from any further consideration. Tier-1 Advisors are reactive 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 tier-1 Advisor which insisted that the victim's limbs be secured. If one were building a FORR-based system to play games, an important tier-1 Advisor would be "if you see an immediately winning move, take it." Only when the first tier of a FORR-based system fails to make a decision does control default to the next tier.

Tier-2 Advisors, in contrast, are not necessarily correct in the full context of the state space. Each of them epitomizes a heuristic, specialized view of reality that can make a valid argument for or against one or more actions. Tier-2 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 of the tier-2 Advisors have an opportunity to comment before any decision is made. The decision they arrive at is the action with highest total strength; this represents a consensus of their opinions. (A tie is broken by random selection.) In a FORR-based game-learning program, a good tier-2 Advisor is "maximize the number of your pieces on the board and minimize those of the other contestant." For the rescue situation, however, tier-2 Advisors are too slow and too risky.

Situation-based behavior has recently been incorporated into FORR as tier 1.5. Each tier-1.5 Advisor has a reactive trigger and a procedure that generates and tests a highly-constrained set of possible solution fragments. A solution fragment emerges from a tier-1.5 Advisor as a sequence of decisions, rather than a single reactive one, a digression from the "sense-compute-execute" loop. This new tier is prioritized like the first, but lacks any guarantee of correctness. A tier-1.5 Advisor triggers when it recognizes that its method may be directly related to the current situation, the way the need to hoist triggers the rescue team's holding devices. Execution of a tier-1.5 Advisor instantiates and tests



acquired useful knowledge
current state
legal actions
A1
Tier 1: Reaction from perfect knowledge
Ak
Decision?
yes
execute decision
no
Ak+1
Tier 1.5: Search and inference triggered by situation recognition
Am
Decision?
yes
no
Tier 2: Heuristic reactions
Am+ 1
Am+ 2
...
An
Voting

Figure l: How FORR makes decisions.

one or more possible solution fragments. Tier 1.5 is prioritized like tier 1, but is not guaranteed correct. The first tier-1.5 Advisor to trigger is ceded control and given limited time to develop a solution fragment. If no tier-1.5 Advisor triggers or produces a sequence of recommended steps, the second tier will make the decision. If a tier-1.5 Advisor constructs a sequence of decisions that it believes relevant, they are executed and then, regardless of the outcome, control is returned to tier 1.

FORR is implemented in Common Lisp. To apply FORR to a domain, one describes the domain, its problem classes, its useful knowledge, and procedures to learn it. (Although learning supports some of the tier-2 Advisors, it is search, not learning, that is the focus of this discussion. Learning is addressed in [Epstein, 1995].) The effectiveness of situation-based Advisors and their role in reasoning is best demonstrated with an example.

3. Ariadne: an Implementation

Ariadne simulates robot path-finding. (Ariadne, daughter of King Minos of Crete, helped Theseus find his way through the labyrinth.) Ariadne is implemented as a set of Common Lisp files that run with FORR. A *problem class* in Ariadne is a particular maze, a rectangular grid with discrete internal obstructions, like the one in Figure 2. A *location* (r, c) is the position in the rth row and cth column of the maze, addressed as if it were an array. In Figure 2 the robot is at (18, 6) and the goal is at (5, 14). *A problem* is to travel from an initial robot location R to some goal location G in a sequence of legal moves, that is, to find a (not necessarily optimal) path to the goal.

Intuitively, a legal move passes through any number of unobstructed locations in a vertical or horizontal line. More formally, a *legal move* is a transition from a state where the robot is at (r, c) to one where it is at (r', c') such that exactly one of the following is true:

- $r = r'$, $c < c'$ and none of $(r, c+1), \ldots, (r, c')$ is obstructed
- $r = r'$, $c > c'$ and none of $(r, c-1), \ldots, (r, c')$ is obstructed
- $c = c$, $r < r'$ and none of $(r+1, c), \ldots, (r', c)$ is obstructed
- $c = c$, $r > r'$ and none of $(r-1, c), \ldots, (r', c)$ is obstructed

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). An Ariadne problem is *solvable* if and only if there exists some path

$<R\ move_1\ loc_1 \ldots loc_{i-1}\ move_i\ loc_i \ldots loc_{p-1}\ move_p\ G>$

such that $move_i$ is a legal move from loc_{i-1} to loc_i for $1 \le i \le p$. The *level of difficulty* of a solvable problem is the minimum value of p for which there is a solution, i.e., the minimum number of legal moves with which the robot can reach the goal. Note that this is different from the Manhattan distance from R to G. Figure 2 is a level 11 problem solved by Ariadne; one solution for it has Manhattan distance 29. Interchanging the robot and the goal produces another level 11 problem in the same problem class.

The robot senses, in any state, its own coordinates, the coordinates of the goal, the dimensions of the maze, and the distance north, south, east, and west to the nearest obstruction or to the goal. The robot does not sense while moving, only before a move. The robot also knows the path it has thus far traversed, and any useful knowledge acquired in previous trips through this maze.

It is important to note that the robot is not given, and does not construct, an explicit, detailed map of the maze. Instead, Ariadne learns descriptive, heuristic information about a particular maze from repeated problem solving in it. This useful knowledge includes gates, dead-ends, and chambers. A *gate* is a location that offers a transition from one quadrant of the maze to another. After each move, Ariadne tests whether its quadrant has changed, 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 gate may not always be helpful; for example, (11, 3) is a gate between quadrants 3 and 2 in Figure 2, but it offers access to little of quadrant 2. Gates are stored in a hash table whose key is the sorted pair of quadrant numbers. 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 when, from the current state, the robot has only one or two moves. Corridors are kept in a hash table whose key is the endpoints. They are enlarged and merged together as necessary. Corridors are a special case of chambers, but they are learned and managed differently.

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 compact, heuristic description at worst overstates the chamber by a bounding rectangle. The *access point* of a chamber is a location within the chamber that affords a view outside it. Figure 2's robot is in a chamber with access point (16, 5) and extent 16 north, 10 east, 20 south, and 4 west; 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. 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.

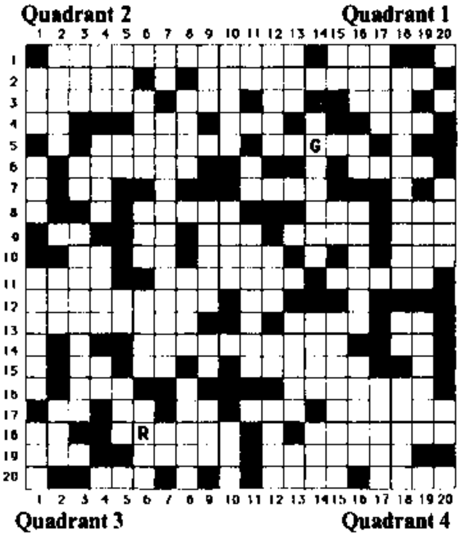


Figure 2: An exploration problem solved by Ariadne. The robot R must move through the grid to the goal G in unidirectional steps through unobstructed locations *without* the map shown here.

Each Ariadne Advisor is listed in Table 1 with a simple description of its rationale. Those in tiers 1 and 1.5 appear in order of their assigned priority. No Way and Victory, the two tier-1 Advisors, are reactive and correct.

The five tier-1.5 Advisors do opportunistic search, and may learn as a consequence. Four of them have triggers that measure lack of progress. Outta Here triggers when the robot's recent locations cover a relatively small fraction of the total area of the maze, or it believes itself to be in a dead-end or chamber not containing the goal. Probe, Super-Quadro, and Wander trigger when the a bounding rectangle on the robot's recent locations is a relatively small fraction of the total maze area. "Recent" and "small" are parameterized. In the experiments described here the values were "the last 30% of the moves have been in no more than 5% of the locations in, or have covered less than 5% of the area of, the maze."

Probe scans to "improve the view," that is, it tries to move the robot in to an access point for the current chamber and then orthogonally leave it. Any new information is recorded as useful knowledge. From (18, 5) in Figure 2, depending upon its recent experience, Probe could generate the path <(16, 5), (16, 3)>.

Outta Here marches out of dead-ends and attempts to exit chambers that do not contain the goal. If in a chamber, Outta Here uses Probe's scan to learn the chamber. If Outta Here

Table J: Ariadne's Advisors for path finding.

	Advisor	Description
Tier 1	No Way	Forbid entry to a dead-end not containing the goal.
	Victory	Take the robot to a visible goal.
Tier 1.5	Outta Here	Seek a pair of moves that minimize the robot's distance to the goal and leave the confined area.
	Probe	Seek to identify and apply an access point for the chamber.
	Super-Quadro	Seek a gate out of the current quadrant, preferably into the goal quadrant.
	Wander	Seek a well-directed, long, L-shaped path with a new view to a thus far unvisited location.
	Roundabout	If the robot is aligned with the goal but there is a wall between them, go around the wall.
Tier 2	Goal Row	Move so that the robot's row coordinate matches that of the goal.
	Goal Column	Move so that the robot's column coordinate matches that of the goal.
	Giant Step	Make the longest possible move, with higher strengths for moves in the direction of the goal.
	Plod	Move one step in any direction, with higher strengths for moves in the direction of the goal.
	Mr. Rogers	Move into the immediate neighborhood of the goal, with higher strengths for closer locations.
	Been There	Object to returning to a location already visited during this problem, with lower strengths for those visited more frequently.
	Done That	Object to moving in a direction already taken from a location already visited in this problem, with lower strengths for directions taken more often.
	Chamberlain	Move into a chamber whose extent indicates that the goal might lie within, and discourage moves into those that do not.
	Quadro	When the robot and the goal are in different quadrants, move to gates that afford access to another quadrant, most strongly for those into the goal quadrant, less strongly for others.
	Opening	Reuse previously successful path beginnings when applicable

were to trigger in Figure 2 when the robot was at (18, 5), it would generate the path <(16, 5), (16, 3)> too.

Super-Quadro scans to change the quadrant. It tries to move the robot in a pair of orthogonal steps to a location whose quadrant is different, preferably the goal quadrant. Super-Quadro would not find a gate from the robot's position in Figure 2.

Wander tests the 8 L-shaped paths that are pairs of longest possible steps in two orthogonal directions, with preference toward those in the direction of the goal. The path it selects ends at a previously unvisited location and makes additional unvisited locations visible. If Wander were to trigger in Figure 2, depending upon its recent experience, it could force the path <(18, 10), (20, 10)>, not a particularly helpful sequence.

Roundabout triggers when the robot is in the same row or column as the goal but cannot see it because of an obstruction. When the robot is so aligned, Roundabout attempts to 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. Roundabout may *fail*, or it may only get closer to the goal than it had been when it started, without actually bringing the goal in sight.

Ariadne's 10 tier-2 Advisors embody path-finding common sense and do no forward search in the problem space. Goal Row and Goal Column attempt to align the robot with the goal. Giant Step and Plod advocate large and small steps, respectively. Mr. Rogers attempts to minimize the Euclidean distance to the goal. Been There and Done That discourage repetition of prior behavior. Chamberlain and Quadro apply learned knowledge about chambers and gates. Opening encourages the reuse of previously successful path beginnings when applicable. (Although such moves may seem odd if the goal is in a different location, the heuristic works well if the old path was successful because it began by moving to an area that offered good access to other parts of the maze.) The simple ideas behind the tier-2 Advisors support rapid computation. Given 10 seconds, no tier-2 Ariadne Advisor has yet run out of time, and no problem has occupied more than a minute.

4. Experimental Results

The data described here arose when Ariadne randomly generated a maze and tested the performance there of different reasoning agents. Because FORR is non-deterministic, results from 10 runs (i.e., 10 randomly-generated mazes) were averaged to produce an *experiment.* Experiments were performed for problems with levels of difficulty 6, 8, 10, and 12 in a 20 x 20 maze that was 30% obstructed. These parameters were selected to provide at least 1000 possible problems at the specified level of difficulty. Effectively, the level of difficulty of a problem is the minimum number of (left or right) turns the robot must make to reach the goal.

In each maze the full version of Ariadne was given 10 learning problems (robot location and goal location pairs) at a fixed level of difficulty. Then 5 newly-generated testing problems for the same maze and level of difficulty were offered to all the agents with learning turned off. A problem of either kind was terminated when the reasoning agent reached the goal or when it had made 100 moves. (The learning problems established a useful knowledge base for those Advisors that depend on it. All agents using such Ad-

Table 2: The performance of a variety of agents after learning in a particular 20 × 20 maze in Ariadne's world. Results are averaged over 10 runs. Path length, moves, and locations are computed only over solved problems. Testing terminated upon solution or after 100 moves. The percentage of the maze explored by breadth-first search on these problems is also shown.

	Agent	Solved	Path Length	Moves	Locations	Triggers	Repetition	BFS %
6-step problems	Reactive	24%	156.2	52.0	29.8	—	42.7	66.1%
	Reactive+	96%	61.2	28.0	22.0	12.8	21.4	
	Reactive-Heuristic	90%	50.4	23.5	16.1	—	31.5	
	FORR	98%	29.7	19.2	15.6	5.9	18.8	
8-step problems	Reactive+	86%	79.7	37.4	29.5	17.7	21.2	87.4%
	Reactive-Heuristic	88%	93.3	37.5	23.7	—	36.8	
	FORR	96%	45.5	30.5	24.3	9.9	20.3	
10-step problems	Reactive+	80%	105.3	50.4	37.8	19.7	25.0	95.2%
	Reactive-Heuristic	66%	132.8	54.3	33.0	—	39.2	
	FORR	86%	60.6	38.3	30.0	15.6	21.7	
12-step problems	Reactive+	64%	118.0	53.8	41.2	29.9	23.4	96.2%
	FORR	80%	69.4	46.2	36.8	25.7	20.3	

visors had equal access to the learned knowledge.) The 100-step limit incorporated any exploration during tier-1.5 search.

Four agents were tested. The *FORR agent* is the full version of Ariadne; it used all 17 of the Advisors to simulate reactive decision making with situation-based behavior. The *Random agent* simply selected random legal moves; this is equivalent to blind search. Three *ablated agents* were formulated by omitting tiers in the hierarchy.

• The *Reactive agent* used only the tier-1 Advisors to simulate correct, reactive decision making alone. If more than one option was left after tier 1, a move was made at random.

• The *Reactive+ agent* used the tier-1 and tier-1.5 Advisors to simulate reactive decision making with situation-based behavior but without heuristic reasoning. If no decision was made after tier 1.5, a move was made at random.

• The *Reactive-Heuristic agent* used the tier-1 and tier-2 Advisors to simulate correct and heuristic reactive decision making without situation-based behavior.

During early trials the Random agent, solved only 12% of 100 level 6 problems and 2% of 100 level 8 problems; it therefore serves merely as a benchmark and was eliminated from subsequent testing. Once any of the other agents performed poorly at some level of difficulty, it too was eliminated from testing at any higher level.

Table 2 reports the results for the ablated agents and FORR averaged across the 10 runs in each experiment. "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 path 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.) "Repetition" measures how repetitive an agent's paths are, calculated as

$$1 \quad \frac{\text{locations}}{\text{moves}}$$

On each problem level, "BFS%" indicates the percentage of the space reachable from the robot's initial position that breadth-first search would have visited on the same test problems. BFS% understates the cost of a physically executed breadth-first search, whose many repetitive subpaths go uncounted here.

The Reactive agent managed to find the goal about a quarter of the time on level 6 problems, not enough to continue testing it at higher levels. Its paths tend to be much more repetitious and several times as long as those of the other agents. The infrequent successes of the Reactive agent occur when large portions of the randomly-generated maze are unreachable from the robot's starting point, the way (1, 20) is in Figure 2. In such a maze the substantial random component of the Reactive agent's behavior was more likely to be effective.

The Reactive-Heuristic agent, FORR's original formulation, began to fail on the level 8 problems. Although it solves about as many problems there as the Reactive+ agent does, its paths are considerably longer and more repetitious, a precursor, in our experience, of dismal performance on slightly harder problems. The Reactive-Heuristic agent was eliminated from testing after level 10.

The situation-based search Advisors of tier 1.5 make a clear contribution when combined with tier 1 as Reactive*, but they have a limited repertoire of behaviors. As the problems become more difficult. Reactive+ is clearly inadequate. It solved only 64% of the level 12 problems.

In summary, as the problems become more difficult, several things happen: breadth-first search reaches an increasing percentage of the accessible unobstructed locations, the situation-based Advisors trigger more frequently, and the ability of the ablated agents to solve the problems becomes markedly inferior. 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. (Indeed, inspection indicates that FORR often "had" the crucial solution fragment to many of its non-successes, but needed a few more steps to finish.) 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.

5. Discussion

This is not a domain in which one would want to do a standard search. The robot's knowledge is so limited that an ordinary evaluation function would be difficult to construct. (For example, closer to the goal is not necessarily better; there may be a very long wall there.) Depth-first search would require fairly elaborate backtracking and loop prevention; very few of the test problems would be solvable in 100 steps with depth-first search. Breadth-first search, while it will always solve the problem, does so at the cost of visiting a high proportion of nodes in the search space and maintaining a very large structure for open paths. Indeed, the data indicate that explicit, breadth-first search in these mazes is nearly exhaustive as the problems become more difficult. Means-ends analysis 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.

Without Tier 1.5, FORR is a reactive system augmented by learned useful knowledge. The results with the Reactive-Heuristic agent, however, simply are not good enough. This agent regularly gets stuck in regions where Giant Step cannot extricate it; it needs maneuvers like Wander's L-shaped path to get out. It also regularly gets close to the goal but cannot see it because of an intervening wall; it needs Roundabout's determined circumvention to get closer.

The situation-based Advisors, however, are insufficient on their own. They consistently trigger more often with Reactive+ than with Ariadne, because most of their triggers measure lack of recent progress, something the robot experiences more often with Reactive*. Tier 1.5 is, effectively, a device to execute subgoals. The subgoal is the opposite of the trigger, e.g., Wander tries to "get out of here," and Roundabout tries to "get around the wall." Subgoals are detected by the program, but their nature is predetermined by the programmer.

There is a complex relationship among the tiers. Tier 1 offers the commonsense inherent in any problem solving task. 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. Another important side effect of the search in tier 1.5 is the acquisition of useful knowledge. (See [Epstein, 1995] for further details.)

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, it is easy to forget that. Tier 1.5 Advisors are kept within the search minimization philosophy in two ways. First, FORR only allocates each Advisor, in any tier, a limited amount of computing time. Solution fragments that take too long to construct will not be 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 were. This constraint saves the tier-1.5 Advisor from a combinatoric explosion. For example, Roundabout follows a procedure that first moves it as close to the wall between it and the goal as possible (the *goal direction)* and then "kicks" it away in an orthogonal direction (the *secondary direction),* with a preference for clockwise. After that, Roundabout seeks the goal, trying to move in the goal direction, then the secondary direction, and then their opposites. Roundabout has some mechanisms to prevent loops in a path, and initially the kick is of length one. As long as Roundabout does not find a way around the wall and still has time left, it will increment the kick by one and try again. *Although this search is quite deep, it also severely curtailed by knowledge; that is why it is effective.*

There are two ways to view Ariadne's task as resource-limited. If CPU time is a scarce resource, then the agent that makes the fewest passes through Figure 1 is best. If traveling time or fuel is a scarce resource, then the agent that constructs the shortest paths is best. On both metrics the full FORR agent achieves a synergy that its individual components lack.

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, one or more potentially relevant cases 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 when resources are very limited.)

Situation-based behavior is not planning either. A plan is a set of actions intended to reach a specific goal [Tate, et al., 1990]. The commander tested holding devices by incorporating them into plans and then simulating those plans until one promised success. The Advisors of tier 1.5, however, 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 and 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 "solved in 100 decision steps."

Situation-based behavior is a resource-grabbing, heuristic digression intended to produce a solution fragment, not a production rule or a macro-operator. Although the trigger of a tier-1.5 Advisor could be the condition of a production rule, the comment generator's response is too complex (particularly since it can be rescinded) to be the action part. 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.

This work has some clear counterparts with Korf s analysis of heuristic search in the tile puzzles [Korf, 1990]. His minimin search "strategy of least commitment" is shared by Ariadne's Plod, but it of necessity lengthens the number of steps in a solution. For example, if the correct, unobstructed move is from (1, 1) to (1, 10), plodding will take 9 moves to get there, although an immediate move to (1, 10) would be legal. Ariadne's Mr. Rogers uses the Euclidean distance heuristic much the way Korf tried heuristic node ordering in the tile puzzles, and with the same disappointing results. Once you get close in this domain, too, there may be better ways to reach the goal. Korf s permission to backtrack with loop prevention is analogous to some of the decision-making in Roundabout. Ariadne has no foolproof loop prevention, but Been There and Done That discourage loops. In Ariadne's graph (rather than tree) search space, Korf indicates that one cannot expect locally optimal solutions. If the search horizon were limited only by how far the robot could see ahead, the Reactive-Heuristic agent should solve few problems, since it sees only one step ahead. That agent's better than expected performance is attributable to its useful knowledge about a particular maze and its general maze-traveling knowledge in the tier-2 Advisors. It is possible to dictate the level of difficulty in Ariadne's problems, but with the tile puzzles there remains some uncertainty about how the level of difficulty impacts upon the ability of the problem solver.

Although Ariadne's maze problems may be reminiscent of recent machine learning work in reinforcement learning, it is important to note that the program's task and fundamental approach are significantly different [Moore and Atkeson, 1993; Sutton, 1990]. Such programs seek convergence to an optimal path through repeated solution of what, according to our definition in Section 3, would be a single problem. In contrast, Ariadne constructs satisficing paths for a set of problems, applying knowledge learned from one problem to the others, instead of from one problem-solving attempt to another attempt at the same problem. The complexity of a maze problem for the reinforcement learners is a function of both goal strength and the number of state-action pairs (the number of reachable locations and directed one-step movements from them). The complexity of a problem for Ariadne, on the other hand, is the numbers of turns required, independent of the size of the maze and the strength of the goal. Memory use is different, too. Ariadne learns abstractions, while the reinforcement learners refine estimates for the value of each one-step move attempted from each state.

Finally, situation-based behavior sheds some light on the ongoing debate about representation and reactivity [Hayes, et al., 1994]. Ariadne's conceptual knowledge includes "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 paper demonstrates that, at least in this domain, the representation of conceptual knowledge is an essential component in a reactive learner.

7. Future Work

An important difference between FORR with tier 1.5 and the rescue team commander is the fact that he ran his successful simulation four times before he implemented it. If Ariadne were to do that, it could substantially shorten its path lengths by removing loops and checking for shortcuts. In the experiments, if a tier-1.5 Advisor discovered a partial solution but could not execute it within the 100-move limit, its agent was still considered to have failed on the problem. This accounts for most of the FORR agent's failures. With path reconstruction, Ariadne would solve even more problems, but not more efficiently, since one would have to count the nodes revisited once the path was refined. The alternative, one we are pursuing now, is to have Ariadne learn additional useful knowledge about past paths.

Based on preliminary empirical evidence, there is every reason to believe that Ariadne will scale up, i.e., that it will continue to perform well in much larger and more tortuous mazes than these. Hoyle, a FORR-based game-learning program, progressed from expertise in spaces with several thousand nodes to spaces with several billion nodes after the addition of only a few tier-2 Advisors. Ariadne has already performed well on preliminary tests in 30 x 30 mazes and continues to improve as we refine its Advisors and its learning algorithms.

Ariadne's continued development addresses the balance between exploration and rapid solution. Since the data indicate that repetitive paths are a forewarning of weakness, we intend to strengthen its penchant for novelty. There are also several new tier 1.5 Advisors on the drawing board.

The thesis of this work is that some carefully constrained search can play an important role in the performance of an otherwise reactive and heuristic system. One could make the task more difficult, say by permitting the goal to shift during the problem, and still expect Ariadne to do well. Preliminary work with Hoyle in the domain of game playing indicates that situation-based behavior does well there too [Epstein and Gelfand, 1995].

Ideally one would like to see a FORR-based program learn tier 1.5, not just use it. *AWL* is an algorithm that learns weights for tier-2 Advisors [Epstein, 1994b]. It was devised to exploit empirical evidence that the accuracy of tier-2

Advisors varies with the problem class. Work is currently under way to apply AWL so that FORR can learn to prioritize tier-1.5 Advisors for each problem class automatically. (Recall that now those priorities are assigned by the programmer.) The next step would be to learn the triggers. We have done some early work on trigger learning with Hoyle [Epstein, and Gelfand, 1995]. The triggers learned there are visual patterns; we expect to analogize some of that work for Ariadne. Finally, one would like to learn the comment-generating procedure to accompany a learned trigger. This may be possible as a generalization over successful solutions after the trigger is satisfied, but it is a difficult problem.

8. Conclusions

For many intractable real-world problems, a suboptimal solution may be acceptable. Situation-based behavior is modeled on human production of suboptimal solutions under time constraints. A reactive system goes from perception (sensing the state of the world) 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 some highly-restricted, intelligent search is an important component in the simulation of efficient, effective 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 and Chapman, 1990] P.E. Agre and D. Chapman. What are Plans for? *Robotics and Autonomous Systems, 6:* 17-34, 1990.

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

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

[Epstein, 1994b] S.L. Epstein. Identifying the Right Reasons: Learning to Filter Decision Makers. In *Proceedings of the AAAI 1994 Fall Symposium on Relevance.,* 68-71, New Orleans, 1994. AAAI.

[Epstein, 1995] S.L. Epstein. On the Roles of Search and Learning in Time-Limited Decision Making. In *Proceedings of the Seventeenth Annual Cognitive Science Conference,* Pittsburgh, 1995.

[Epstein and Gelfand, 1995] S.L. Epstein and J. Gelfand. Learning Spatially-Oriented Game-Playing Agents through Experience. In *Proceedings of the Seventeenth Annual Cognitive Science Conference,* Pittsburgh, 1995.

[Hayes, Ford and Agnew, 1994] P.J. Hayes, K.M. Ford and N. Agnew. On Babies and Bathwater: A Cautionary Tale. *AI Magazine,* 75(4): 15-26, 1994.

[Klein and Calderwood, 1991] G.A. Klein and R. Calderwood. Decision Models: Some Lessons from the Field. *IEEE Transactions on Systems, Man, and Cybernetics, 21(5):* 1018-1026, 199L

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

[Korf, 1990] R.E. Korf. Real-Time Heuristic Search. *Artificial Intelligence, 42(2-3):* 189-211, 1990.

[Maes and Brooks, 1990] P. Maes and R. A. Brooks. Learning to Coordinate Behaviors. In *Proceedings of the Eighth National Conference on Artificial Intelligence,* 796-802, Boston, MA,' 1990. AAAI Press.

[Moore and Atkeson, 1993] A. W. Moore and C. G. Atkeson. Prioritized Sweeping: Reinforcement Learning with Less Data and Less Time. *Machine Learning, 13(1):* 103-130, 1993.

[Sutton, 1990] R.S. Sutton. Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. In *Proceedings of the Seventh International Conference on Machine Learning,* 216-224, Austin, TX, 1990. Morgan Kaufmann.

[Tate, Hendler and Drummond, 1990] A. Tate, J. Hendler and M. Drummond. A Review of AI Planning Techniques. In J. Allen, J. Hendler, & A. Tate (Ed.), *Readings in Planning,* 26-49 26-49, San Mateo, CA, 1990. Morgan Kaufmann.