Representation and Reasoning for Pragmatic Navigation

Susan L. Epstein

Department of Computer Science
Hunter College of The City University of New York
New York, NY 10021
epstein@roz.hunter.cuny.edu

Abstract

This paper focuses upon features of two-dimensional space that facilitate and/or obstruct travel through it. Three kinds of facilitators and four kinds of obstructers are identified. For each one, learning algorithms and navigational applications are proposed. The program described here simulates a robot that learns such features as it encounters them, and then applies them to subsequent travel decisions. The resulting system travels pragmatically through a variety of two-dimensional worlds.

1. Introduction

As we send robots into dangerous, distant, or dynamic domains, their ability to navigate through them becomes an increasingly important issue. Because the robots' controllers may have no map for the territory to be traversed, a reasonable approach would be to have the robot learn about the space it travels through. There are two issues that arise, however, if we require the robot to map this new space explicitly. First, accurate mapping is unlikely, since robotic sensors are subject to noise and robotic travel is subject to error. Second, unless people plan to follow the robot into the territory, the robot's principle goal is to travel from one point to another, not to construct a map. The alternative proposed here is to satisfice, to have the robot learn approximations of the territory that support competent, rather than optimal, travel. The foundation concepts for this work are the facilitation and the obstruction of movement through space.

Pragmatic navigation aspires to provide robust, competent performance in two-dimensional space, performance that improves across time and is resilient to changes in origin and destination. Instead of pre-engineered expertise for a particular territory, a pragmatic navigator learns its way around a new territory from a series of trips through the territory, trips whose origins and destinations differ. Just as a person who travels efficiently through a campus does not retain a detailed description of each trip or remember the location of each tree and rock, a pragmatic navigator ignores much travel history and many topographical details. Instead of a detailed map, pragmatic navigation relies upon features that support efficient travel or make it more difficult, such as a door into a room or an extended wall. In a new territory, a pragmatic navigator initially performs as a competent novice, and then, as it learns features of the territory from traveling there, it uses that knowledge to improve its performance. As envisioned here, most features are heuristic rather than absolutely accurate; they are useful approximations that describe a territory and travel experience through it. Such representation deliberately sacrifices detail in exchange for efficient storage, retrieval, and computation.

The pragmatic navigation task was first suggested as a problem that would challenge the power of search algorithms (Korf 1990). The robot's world is a *maze*, a discrete, rectangular grid with external walls and internal obstructions like the 14 × 14 maze that is 30% obstructed in Figure 1. (All the examples in this paper are taken from actual runs; the experiments themselves are on substantially larger

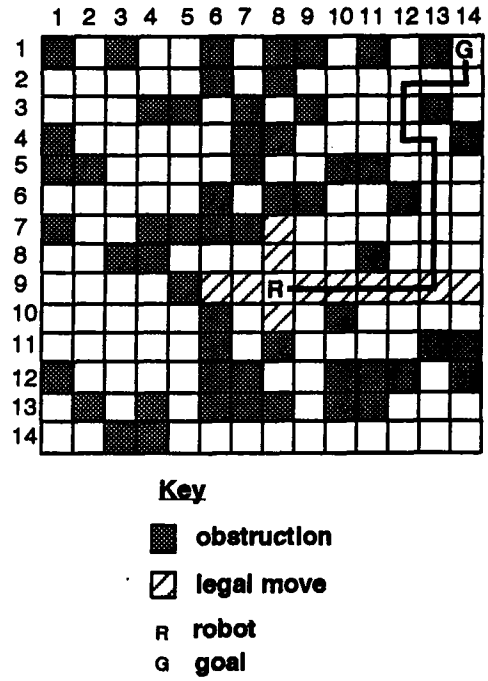

Figure 1: A problem and its solution in a maze.

mazes than this one.) 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. 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. In Figure 1 the problem is to travel from (9, 8) to (1, 14). In any state, the robot senses only 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 knows the path it has thus far traversed in the current problem, but it is not given, and does not construct, an explicit, detailed map of the maze like the one in Figure 1.

Instead of a map, pragmatic navigation relies upon two kinds of features to describe a territory: those that support efficient travel (*facilitators*) and those make it more difficult (*obstructers*). These features constitute *useful knowledge*, and are learned for a specific territory from experience. Although it may be approximate, useful knowledge is expected to enhance performance. The robot remembers any useful knowledge acquired in previous trips through this maze, perhaps the lengthy irregular wall separating the eastern and western portions of Figure 1, and the dead-end at (12, 13). Each feature is prespecified by the system designer, including its learning algorithm, learning time limit, and learning schedule (after a decision, a problem, or a set of problems).

Intuitively, a legal move passes through any number of unobstructed locations in a vertical or horizontal line. The robot in Figure 1 has 11 legal moves: north to (7, 8) and (8, 8), east to (9, 9) through (9, 14), south to (10, 8), and west to (9, 6) and (9, 7). A problem is *solvable* if and only if there exists some path
$<R = \mathrm{loc}_0 \mathrm{move}_1 \mathrm{loc}_1 \ldots \mathrm{move}_i \mathrm{loc}_i \ldots \mathrm{loc}_{p-1} \mathrm{move}_p \mathrm{loc}_p = G>$
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. (Effectively, the level of difficulty of a problem is one more than the minimum number of left or right turns the robot must make to reach the goal.) This is different from the Manhattan distance from R to G. Figure 1 is a level-6 problem; one six-move solution for it, with Manhattan distance 16, is indicated there. Interchanging the robot and the goal produces another problem at the same level. The *heading* of the task is the subset of {north, east, south, west} that describes the direction from R to G. The heading of the task in Figure 1 is {north, east}.

This task has several performance criteria. The robot is expected to solve multiple problems in the same maze. Easier problems should be easier to solve. The robot is expected to perform quickly. Speed can be measured as elapsed computation time, number of decisions, or path length (Manhattan distance) traveled. The robot is also expected to perform efficiently. Efficiency can be measured as the number of distinct locations visited or the percentage of repeated locations in a path. Finally, the robot is expected to learn; its performance should improve with experience.

The task formulated above is not amenable to traditional AI techniques. Depth-first search requires substantial backtracking; its paths are long and repetitive. Breadth-first search visits too many nodes; on most hard problems it approaches exhaustive search while it visits a high proportion of nodes in the search space and maintains a very large structure for open paths. Means-ends analysis is inapplicable because it requires knowledge about the vicinity of the goal to reason backwards. Best-first search with a "sensible" evaluation function, such as Euclidean distance to the goal, is easily misled by *deceptive* problems where proximity is not a valid indicator of progress. (The goal might be hidden behind a long wall so that the robot must move away from the goal to reach it eventually. For example, placing the robot at (9, 4) and the goal at (8, 5) in Figure 1 produces a deceptive level-6 problem.) For a large maze, explicit search would be extremely inefficient, perhaps intractable.

The next two sections describe the facilitators and obstructers used by a program called *Ariadne*, whose reasoning process is described in Section 4. (Ariadne, daughter of King Minos of Crete, told Theseus how to find his way through the labyrinth that protected a great treasure.) Section 5 evaluates the role of facilitators and obstructers in Ariadne. Subsequent sections consider Ariadne's cognitive plausibility and related work.

2. Facilitators

Ariadne has three kinds of facilitators: gates, bases, and corners. A gate has, in theory, the ability to provide a transition from one large segment of space to another. A base repeatedly appears as a counter-intuitive choice in successful paths. A corner offers the possibility of a new direction.

Since the robot knows the dimensions of the maze, it can calculate quadrants. A *gate* is a location that offers a transition from one quadrant of the maze to another. After each move, the robot can test 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, (8, 10) is a gate between quadrants 3 and 4 in Figure 2(a), but it offers access to little of quadrant 3. Each gate is stored with its *extent* (rectangular approximation of the locations from which it can be reached) in a hash table whose key is the sorted pair of quadrant numbers. The extent of (8, 10) in Figure 2(a), for example, is the rectangle with vertices (6, 10), (9, 10), (9, 5), and (6, 5). Ariadne only learns the gates it visits, so many locations in Figure 2(a) that satisfy the definition of gate were not learned and are not marked. The subdivision of the maze into only four areas (the quadrants) was deliberate. If one specifies more, there are an increased number of gate categories to manage (as many as ${}_nC_2$ for n areas). If one specifies fewer, there is too little transition information.

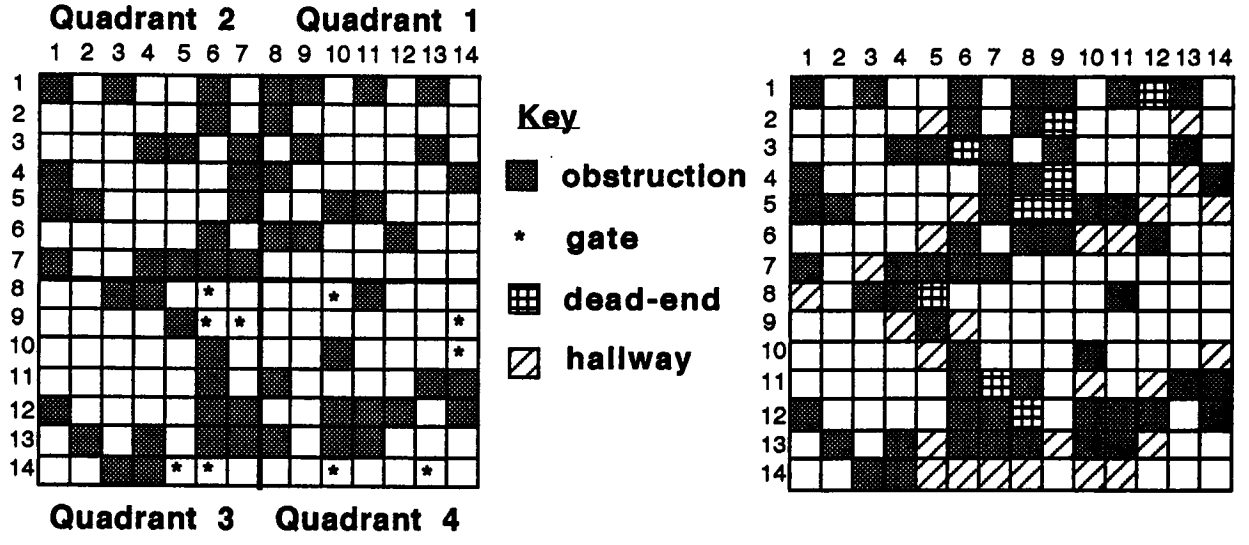

Figure 2: After 20 problems: (a) gates and (b) corridors learned for a simple maze.

A *corridor* is a passageway of width one that either has a single exit (a *dead-end*) or is a *hallway*. A *pipe* is a straight hallway, that is, its endpoints lie in the same row or the same column. In Figure 2(b), there is a hallway from (13, 5) to (14, 8) and a pipe from (14, 10) to (14, 11). Some pipes offer a view of the space after their far end. For example, from (14, 9) in Figure 2(b) the robot can move not only to the ends of the pipe from (14, 10) to (14, 11), but also beyond it to (14, 12). Other pipes do not offer a view of the space after their far end, but since a pipe is not a dead-end, that promises a turn in some other direction. For example, from (4, 10) in Figure 2(b) the robot can see both ends of the (length one) pipe at (4, 13) but not beyond it; that promises a turn at the far end. Such a turn-promising far end of a pipe is called a *corner*. A corridor is learned when, from the current state, the robot has only one or two moves. The two endpoints of a corridor serve as the keys to a hash table which also indicates whether or not they are for a dead-end. Corridors are enlarged and merged together as necessary. Like gates, only corridors that Ariadne experiences are learned.

A *base* is a location in a maze that appears to have been a key to a successful path. In the author's home town, people regularly give directions beginning "first you go to the Claremont Diner." Although it served memorable cheesecake, the Claremont Diner burned down 15 years ago, and there is nothing particularly significant about the car dealership that has replaced it. What is significant is that the diner was at a location that affords ready (not necessarily shortest path) access to other locations in a 10-mile radius. A base is such a location. Bases are learned after a successful problem; the algorithm first corrects the path to eliminate loops and unnecessary digressions. A base is a location in that corrected path that was not in the heading from *R* to *G*. A base is not a dead-end or *G* itself, and a path intended

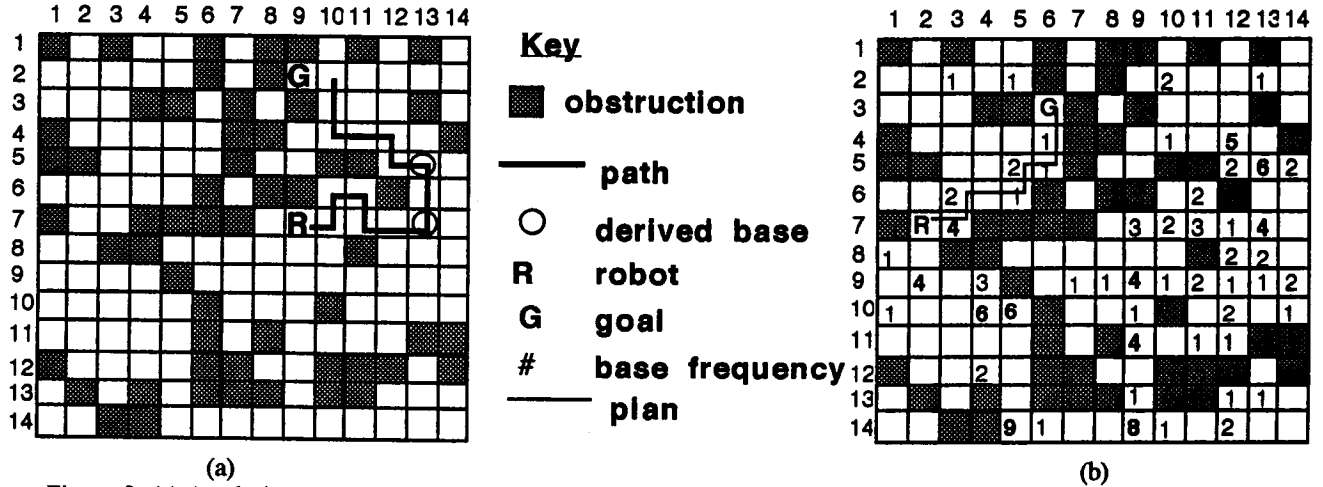

Figure 3: (a) A solution path and the bases that arise from it. (b) A plan for a problem formulated from learned bases.

to circumvent a wall contributes only its most extreme positions opposite the original headings. Bases are stored on a list with their *frequency*, the number of times they have been identified in different problems. The bases learned from one solution path are circled in Figure 3(a), where the heading was {north} and the eastern-most corners became bases.

Bases facilitate some primitive, high-level planning for Ariadne's decision making. A *plan* in Ariadne is a sequence $< b_0\ b_1\ b_2\ \ldots\ b_{i-1}\ b_i\ b_{i+1}\ \ldots\ b_n\ b_{n+1} >$ where b_0 is the robot's current location, b_{n+1} is the goal, b_i is a base for $i = 1$ to n, b_i is aligned vertically or horizontally with b_{i+1}, and either b_{i-1} is closer to b_0 than b_i is, or else b_{i+1} is closer to G than b_i is. Plans are constructed with bidirectional search on aligned bases, with preference for bases of higher frequency, as if there were no obstructions. A plan fails when the robot is at some b_i and there is an intervening obstruction that prevents its move to b_{i+1}. An example of a plan Ariadne formulated for a problem is shown in Figure 3(b), where the bases learned after 20 level-6 problems in the same maze are indicated by their frequency values. In Figure 3(b), some bases, such as (14, 5) and (14, 9) are the keys to the only route between the eastern and western portions of the maze. Others, such as (7, 13) and (9, 9) lie at important intersections, like the Claremont Diner.

3. Obstructers

Ariadne identifies four kinds of obstructers: certain corridors, chambers, bottles, and barriers. A corridor may obstruct movement either because it is a dead-end, such as (2, 9) in Figure 2(b), or because it is a pipe, whose internal locations afford access only to each other. Chambers and bottles are circumscribing rectangular approximations of restricted spaces that are less narrow than a corridor, and have one or more exits. A barrier is a linear approximation of contiguous obstructed positions. Learning an obstructer is often triggered when the robot has been hampered in its ability to move through space. For example: "The area of the territory covered by the last 30% of the moves was less than 10% of the total maze area or included less than 10% of the possible maze locations," or "All the legal moves have been visited at least once," or "60% of the decision limit has been exhausted."

A *chamber* is an irregularly shaped space with an access point and an approximate extent. The extent is a bounding rectangle, a compact, heuristic approximation of the furthest in each direction one can go in the chamber. The *access point* is a location within the chamber that affords a view outside it, but need not be on the border of the extent. Figure 4(a) shows a chamber with extent 1 north, 13 east, 5 south, and 9 west. When the robot moved to access point (4, 13) it saw beyond that extent to the south. All locations reachable from the robot's initial position really constitute one large chamber, but the chambers that Ariadne learns are more limited and room-like. The learning algorithm for a chamber is triggered when the problem has been underway for some time, the robot has been recently constrained and has been in its current location before, and there are few legal moves to locations not yet visited on this problem or the goal is remote. The learning algorithm for a chamber first estimates the dimensions according to its current sensing, and then tries to move to a location where the chamber appears both higher and wider. From its current location the robot scans once horizontally, and then from the scanned location offering the largest view scans once again vertically. (If there are no horizontally-adjacent legal locations, the vertical scan is performed first.) If the procedure identifies a sequence of one or two locations (access points) that enlarge the extent, at least one of which is previously unvisited during this problem, it records the chamber's extent and access point on a list. The scan for the chamber in Figure 4(a) began from (4, 12). 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.

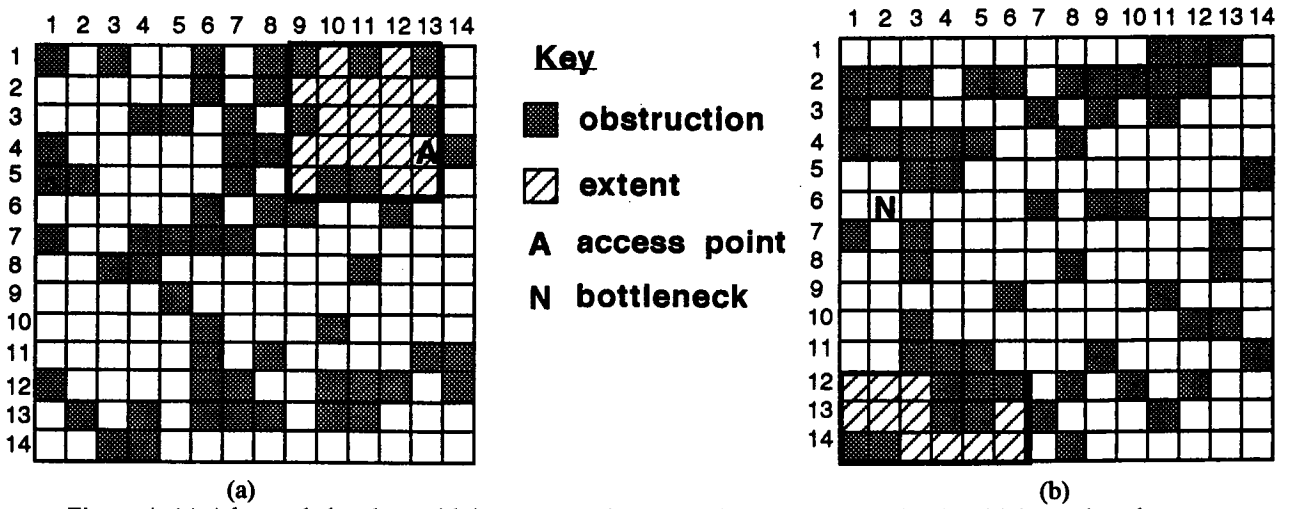

Figure 4: (a) A learned chamber, with its extent and access point (b) A learned bottle with its neck and extent.

A *bottle* is another useful knowledge description of a constrained space, similar to a chamber. Chambers, however, are learned deliberately by search, whereas bottles are learned without search from analysis of the entire path after a problem is completed. A potential bottle begins with a location that was visited more than once, and is repeatedly extended in both directions along the path by immediately neighboring positions only if it includes several spots, is not corridor-like, and does not ultimately encompass more than 15% of the area of the maze. Once a bottle is identified and its extent computed, its *neck* (not necessarily contiguous entry and/or exit point) is identified. Bottles are stored in a hash table as an extent and a neck. Figure 4(b) shows a bottle with extent 12 north, 6 east, 14 south, and 1 west, and neck (6, 2).

A *barrier* is a linear approximation of a wall that obstructs movement. Figure 5 shows the barriers learned after 20 problems in a simple maze. The barrier from (13, 10) to (11, 13), for example, is an approximation of the wall in the lower right corner of the maze. Barriers are learned different ways, depending upon the search context in which they arise. In each case, the learning algorithm is a function of the preferences in the rationale that produced the search path. Experience determines which barriers are encountered; thus there could have been some barriers learned to describe the irregular vertical wall between the eastern and western portions of the maze in Figure 5, had the program encountered difficulty when required to avoid them. Each retained barrier is an object with endpoints, slope, intercept, and length.

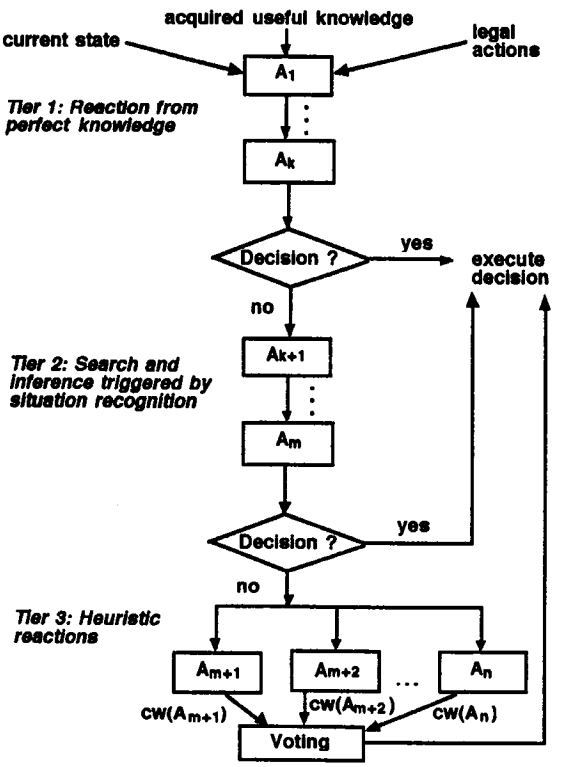

Figure 6: How FORR makes decisions.

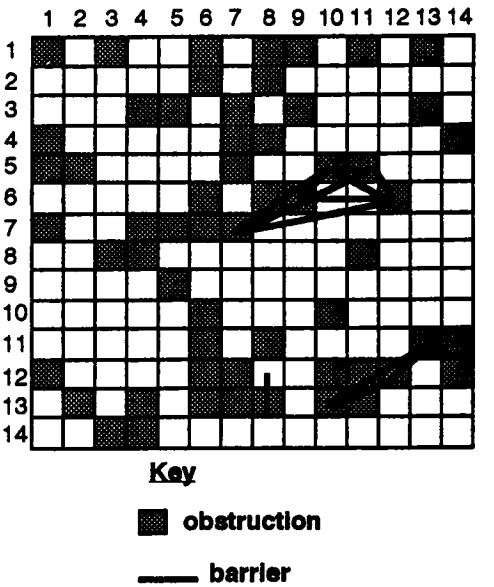

Figure 5: After 20 problems, barriers computed for a simple maze.

4. Reasoning for Pragmatic Navigation

FORR (FOr the Right Reasons) is a problem-solving and learning architecture that models the transition from general expertise to specific expertise, and capitalizes on methods that people use (Epstein 1994). FORR approaches problem solving as a sequence of reasonable decisions based on the current state of the world and learned useful knowledge. Ariadne is a FORR-based system for pragmatic navigation; it applies facilitators and obstructers to the maze task. FORR's three-tier hierarchical model of the reasoning process (shown in Figure 6) mediates among decision-making rationales called *Advisors*. Each Advisor is a "right reason," implemented as a time-limited procedure. Input to each of Ariadne's Advisors is the current state of the world, the current permissible actions from that state, and any learned useful knowledge about the current territory. Each Advisor outputs any number of *comments* that support or discourage permissible actions. A comment lists the Advisor, the action commented upon, and a *strength*, an integer in [0, 10] that measures the intensity and direction of the Advisor's opinion. Details on the architecture appear in (Epstein 1995). A full listing of Ariadne's 30 Advisors and the useful knowledge 18 of them apply appears in Table 1.

The tiers are consulted in turn; if any tier makes a decision, it is executed and control is returned to tier 1. Advisors in tiers 1 and 2 are consulted in the predetermined, fixed order shown in Table 1, and may make a unilateral decision. The four tier-1 Advisors are perfectly correct, reactive procedures that decide quickly. Tier-1 Advisors also have the ability to veto a move, eliminating it from further consideration. The eight tier-2 Advisors are situation-based procedures that do time-limited search in an attempt to produce a sequence of moves that they then mandate. Each

Table 1: Ariadne's Advisors with tiers 1 and 2 in prioritized order.

Tier	Advisor	Rationale	Useful knowledge
1	*Victory*	If goal is reachable by a legal move, go there.	—
1	*Can Do*	Move adjacent to goal.	—
1	*No Way*	Avoid dead-ends.	Corridors
1	*Pipeline*	Avoid internal locations in straight corridors.	Corridors
2	*Roundabout*	Circumnavigate intervening obstructions.	Barriers, corridors
2	*Outta Here*	Exit chamber or dead-end not containing goal.	Chambers, corridors
2	*Probe*	Determine the current extent and try to leave it.	—
2	*Patchwork*	Repair plans.	Bases
2	*Other Side*	Move robot to opposite side of goal.	—
2	*Super Quadro*	Search for entry into goal's quadrant or into new quadrant.	Gates
2	*Wander*	Move far in an L-shaped path.	Barriers, average problem steps, bases, corridors
2	*Humpty Dumpty*	Seek barriers.	—
3	*Adventure*	Move to thus far unvisited locations, preferably toward goal.	Barriers
3	*Been There*	Discourage returning to location already visited during this problem .	—
3	*Chamberlain*	Move into chamber that contains goal; avoid chamber that does not.	Chambers
3	*Contract*	Take large steps when far from goal, and small steps when close to it.	—
3	*Cork*	Move into a bottle that contains goal; avoid a bottle that does not.	Bottles
3	*Corner*	Move to the end of a pipe that promises a turn.	Corridors
3	*Crook*	Move to the end of a crooked corridor.	Corridors
3	*Cycle Breaker*	Stop repeated visits to same few spots.	—
3	*Detour*	Move away from barriers that obstruct goal.	Barriers
3	*Done That*	Discourage moves in same direction as before from a previously-visited location.	—
3	*Giant Step*	If recently confined, take a long step, preferably toward goal.	—
3	*Goal Column*	Align robot horizontally with goal, if it is not already.	—
3	*Goal Row*	Align robot vertically with goal, if it is not already.	—
3	*Home Run*	Move toward bases.	Bases
3	*Hurry*	Take big steps early and small steps late.	Average problem steps
3	*Leap Frog*	Execute opportunistic plans.	Bases
3	*Mr. Rogers*	Move into neighborhood of goal.	—
3	*Opening*	Begin as a previously successful path did.	Openings
3	*Plod*	Take a one-unit step, preferably toward goal.	—
3	*Quadro*	Move into goal's quadrant or into new quadrant.	Gates

tier-2 Advisor has a trigger that signals its applicability and a search method that attempts to compute solution fragments (sequences of moves) to address the identified situation. The solution fragments generated by a tier-2 Advisor are tested serially. The 20 tier-3 Advisors are reactive, time-limited heuristics that embody path-finding commonsense and do no search in the maze. Each may recommend or oppose any number of unvetoed legal moves. All tier-3 Advisors vote together, and the simple ideas behind them support rapid computation. Given 10 seconds, none has ever run out of time on level 10 problems.

Advisors reason with facilitators by attempting to exploit them. Super Quadro and Quadro apply gates to move the robot into the goal's quadrant or to change quadrants, the former with limited search, the latter with a reactive heuristic. Four Advisors apply bases, and plans constructed from them, to revisit key locations: Patchwork repairs plans, Wander makes L-shaped paths into unknown territory in inverse proportion to the percentage of locations known to be bases, and Home Run and Leap Frog react to plans. Corner and Crook apply corridors. Corner advocates a move to the far end of a pipe that turns, more strongly when the pipe is in the direction of the goal, less so when it is opposite the direction of the goal. Crook advocates a move to the near end of a corridor that is neither straight nor a dead-end, again more strongly when the corridor is in the direction of the goal, less so when the far end of the corridor is opposite the direction of the goal.

Some Advisors simply avoid obstructers. No Way vetoes any move that would lead the robot into a dead-end that could not contain the goal, and Pipeline vetoes a move to any location within a pipe which it can see through. For example, in Figure 2(b) from (14, 9) Pipeline would veto moves to (14, 10) and (14, 11); if traveling east efficiently, one would go through that pipe, not into it. Outta Here, Roundabout, and Wander all avoid dead-ends during search. Roundabout and Wander restrict their search to avoid known barriers. Adventure and Detour avoid moves that position the robot with a known barrier between it and

the goal. If a barrier already intervenes, Detour also encourages moves to avoid it.

Other Advisors must consider whether an obstructer might contain the goal. Outta Here applies chambers with restricted search to exit a chamber if the goal is not there, and enter it if the goal might be there. Probe learns chambers as a side effect of limited search. Chamberlain is a reactive heuristic version of Outta Here, and Cork is the bottle version of Chamberlain. Thus when the robot is outside the bottle, Cork discourages moves into a bottle whose extent indicates that it cannot contain the goal, and encourages moves into the neck of a bottle whose extent indicates that it can contain the goal. When the robot is inside the bottle, Cork reverses this advice.

Finally, two Advisors apply useful knowledge supplied by default by FORR, knowledge that is not specific to pragmatic navigation: 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 average number of problem steps is also referenced by Hurry and by the triggers of several tier-2 Advisors.

5. Empirical Design and Results

Pragmatic navigation is achieved by the execution of FORR's Figure 6 decision process with all the Advisors of Table 1. Together Ariadne's Advisors keep the robot on a reasonable path toward the goal, exploiting the facilitators and steering clear of the obstructers. To demonstrate that it is facilitators and obstructers that empower Ariadne, we devised an ablation experiment consisting of 10 runs. On each run, Ariadne generated a 20 × 20 random maze that was 30% obstructed, and solved 20 learning problems there. Then learning was turned off, and 10 newly-generated testing problems for the same maze were offered both to the full version of Ariadne and to a *No-Learn agent* that applied all the Advisors but made no useful knowledge available. This experiment was repeated for problems at levels 4, 6, and 8. The learning problems established a useful knowledge base for those Advisors that depend on it. A problem of either kind was terminated when the reasoning agent reached the goal or when it reached the decision step limit which included all exploration during tier-2 search. On levels 4 and 6, this limit was set to 200. Above level 6, it was set to 1000 to give the agents ample opportunity to solve each problem. The higher limit permitted more experience, so that Ariadne acquired additional useful knowledge to support better comments.

Table 2 reports the results for No-Learn and Ariadne averaged across the 10 runs in each experiment. In Table 2, "distance" is the Manhattan distance along the path to the goal. Since a step may move through one or more locations, path length varies among problems of the same difficulty. "Decisions" is the number of steps taken during tier-2 search or solution, while "moves" is the number of steps in the solution. (The distinction between the number of decisions and number of moves is important: a *move* appears in the solution path, while a *decision* is a step taken during tier-2 search or solution.) The number of distinct locations actually visited during those moves is reported as "locations." Distance, moves, and locations are computed only over solved problems. (This tends to make the ablated agents look somewhat better than they actually are, because they solve the easier problems.) "Time" is execution time per testing problem, in seconds.

On every level Ariadne solved a few problems that No-Learn did not. By level 6, No-Learn had fairly long paths, but consumed similar time and decision cycles. On level 8, however, its inadequacy was measurable at the 95% confidence level. Although No-Learn managed to solve 93% of the problems there, inspection indicated that it had relied heavily on tier 2, that is, had substituted search for knowledge. As a result, its paths were far longer, and its decision cycles and solution time substantially greater. The degree of repetition in the paths is also indicative of their behavior. On level 8 No-Learn visited 36% of its locations more than once while Ariadne revisited only 8%. Clearly facilitators and obstructers enhance Ariadne's performance.

Nor is there apparently any danger of learning so much useful knowledge that a detailed, grid-like map of the maze would have been a more efficient representation than Ariadne's learned heuristic features. After 20 level-8 problems in 20 × 20 random mazes, for example, there were on average only 41.8 barriers, 83.6 bases, 1.0 bottles, 3.7 chambers, 49.3 corridors, and 18.2 gates. As a graph, however, such a maze would have 280 nodes and approximately 1485 edges.

Table 2: The performance of Ariadne and an ablated version of it after learning in a particular 20 × 20 maze in Ariadne's world. Results are averaged over runs in 10 mazes.

Problem level	Agent	Distance	Decisions	Moves	Locations	Time
4	No-Learn	23.77	35.27	11.99	10.17	1.41
4	Ariadne	18.74	20.33	11.64	10.95	1.21
6	No-Learn	40.33	58.53	21.77	18.02	2.38
6	Ariadne	27.23	47.61	16.22	15.15	1.99
8	No-Learn	99.08	202.85	44.10	28.39	7.08
8	Ariadne	38.15	115.19	24.16	22.19	4.00

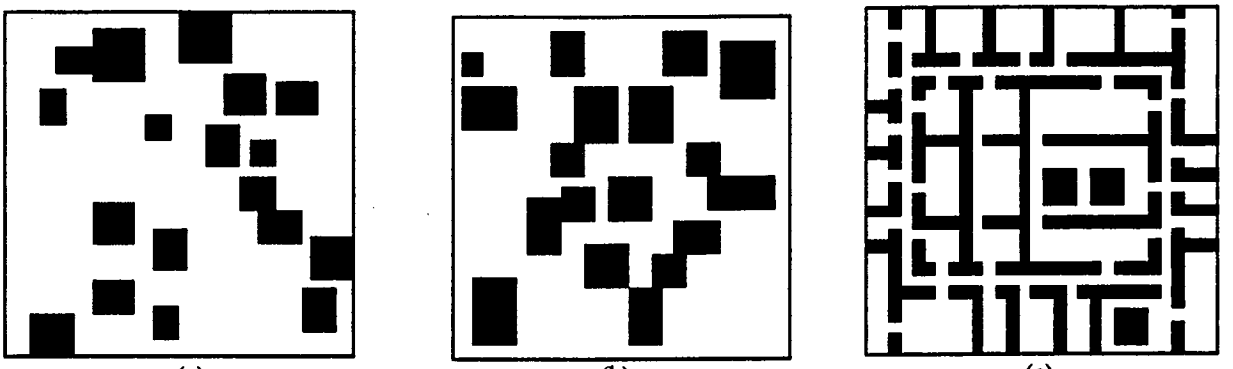

Figure 7: Some non-random environments: (a) warehouse (b) furnished room (c) office. Grids are omitted for clarity.

To measure the efficacy of Ariadne's problem solving, we compare it here with two standard AI techniques: breadth-first search and heuristic search with an evaluation equal to the Euclidean distance from the robot to the goal. In a separate experiment, again averaging 10 runs on level 8, on the testing problems Ariadne only visited 7.93% of the locations in the maze, while breadth-first search visited 79.89%. (This somewhat understates the cost of a physically executed breadth-first search, whose many repetitive subpaths go uncounted here.) In yet another 10-run experiment, this time at level 10, best-first search was able to solve only 37% of the problems within 1000 steps, and averaged path lengths of 92.32 on these solved problems, versus Ariadne's 55.30 path length with a 93% success rate. Best-first search averaged 271.23 seconds per problem, Ariadne 5.93 seconds.

Although random mazes present interesting challenges, real navigators face environments which are not random. To test Ariadne's robustness, three other classes of mazes intended to model more realistic worlds were constructed. These maze classes represent furnished rooms, warehouses, and office suites on a 40 × 40 grid. An example of each appears in Figure 7. A *warehouse maze* models a single room, much like the typical basement, garage, attic or storeroom, with non-overlapping rectangular obstructions scattered about. A *furnished room maze* models a room some of whose rectangular obstructions may overlap or be balanced along the perimeter. An *office maze* models a single floor in an office building, with an outer rectangle of offices (those that could have windows) bordered within by a rectangular hallway and corner offices accessible only through an adjacent office.

Ariadne was developed for random mazes. Without any changes in the useful knowledge or the Advisors, we tested the program in mazes like those in Figure 7. Level-8 problems were run on the offices and warehouses, but it was difficult to find a furnished room with enough problems at any higher level than 4, apparently because the required gap between the perimeter and the furniture provides ready access to most locations. Although these new mazes were 4 times larger than those in earlier experiments, the decision-step limit remained at 1000. Ariadne solved all the furnished room problems easily, 54% of them as well or better than the problem generator's solution. (This is possible because the problem generator minimizes the number of turns on the path to the goal, not the distance. In some mazes, a path with more turns can be shorter.) It also solved all the warehouse problems readily, 39% as well or better than the problem generator. The office mazes presented a greater challenge. Only 9% of Ariadne's solutions were as good or better than the problem generator's, and in several cases Ariadne did not solve the test problem within the 1000 decision-step limit. (In contrast, the program devoted an average of 37.47 decisions to furnished room problems, and 64.65 to warehouse problems.) Inspection indicated that in most cases the solution was near at hand, but the corner offices had been sufficiently deceptive to demand additional search.

6. Cognitive plausibility

There is no claim here that Ariadne is a cognitive model of a human navigator, only that many of its features are cognitively plausible. Several of the Advisors do model principles of naive geographic reasoning (Egenhofer & Mark 1995) all of us readily recognize: Plod's tentativeness, Adventure's curiosity, and Hurry's anxiety. In addition, cognitive scientists have found evidence for Contract's rationale (Golledge 1995) and the rationales of several of the other Advisors (Gryl 1995). There have, however, been no tests of human subjects on problems as difficult as these. Indeed, psychologists doubt that most people with the same visual limitations as Ariadne's robot could solve problems on this scale much above level 4 (personal communication, Ratterman).

The use of a learning architecture, rather than a prespecified one, is supported by evidence that people evolve superior performance. Human expertise develops only from repeated experience at problem solving (Ericsson & Charness 1994; Ericsson, Krampe, & Tesch-Römer 1993; Ericsson & Smith 1991; Piaget & Inhelder 1967). This expertise is applicable to a related set of problem classes. For example, an expert path finder in unfamiliar territory will immediately wonder about dead-ends and not about the color of obstructions. Experts rely upon a foundation of domain knowledge that provides a source of focus and direction to provide a

baseline level of competence. Thus the architecture need not begin with total ignorance; it is reasonable to provide it with general domain knowledge and the methods to specialize it, for example, by learning feature instances.

Ariadne's facilitators and obstructers form its *cognitive map*, its representation of its world. These features are consistent with the ways humans represent space. People use constructed representations of the visual world to make inferences about space, representations that are based upon, but more abstract and general than, perception (Tversky 1991). These representations systematically distort visual perception to facilitate recall (Tversky 1992). They are integrated with many other kinds of information to form a model that the human user does not require to be complete or consistent. Ariadne's reliance upon multiple representations and its tolerance for inconsistent and even incorrect information, geographers tell us, is much like the naive geography that people rely on (Egenhofer & Mark 1995). Even the use of levels (number of turns) for degree of problem difficulty is supported by results that people gauge distance that way (Sadalla & Magel 1980).

7. Related Work

Although Ariadne's maze problems may be reminiscent of recent work in reinforcement learning, it is important to note that the program's task and fundamental approach are significantly different (Moore & Atkeson 1993; Sutton 1990). Such programs seek convergence to an optimal path through repeated solution of what, according to our definition, would be a single problem. In contrast, Ariadne has no mechanism that would guarantee optimality, and will quickly settle upon the same route in most cases. 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.

Another suggested learning approach was a case-based planning method for the grid world that operated in a set of abstraction spaces, and stored both detailed and abstracted solution paths (Branting & Aha 1995). These mazes were somewhat simpler versions of Ariadne's warehouses, which present few dead-ends, narrow-necked chambers or bottles, or effective barriers between large regions. Once again, Ariadne would find them quite easy.

One way to characterize an approach to learning navigation for a robot is by the degree to which it reflects what is known about human perception and behavior. PLAN is a connectionist model that integrates path finding into general cognition (Chown, Kaplan, & Kortenkamp 1995). PLAN learns a topological map of experienced landmarks, a route map of place sequences and directions between them, and a survey map of global information. All three maps are based on a visual scene rather than an aerial view. The resultant system has strong biases to human sensory orientation (particularly in its refusal to see more than 180° about its position) and limited short-term memory, neither of which is necessary for intelligent robot navigation. The authors claim that such maps are therefore based on experience, but the experience they record is purely visual or rote experience. In contrast, Ariadne retains knowledge that represents both momentary experience (e.g., a gate) and reasoning about a sequence of experiences (e.g., a base). Although developed independently, some of PLAN's features are quite similar to Ariadne's: its gateways are similar to Ariadne's gates, its regions similar to Ariadne's bottles and chambers. PLAN makes its global overview explicit, and plans hierarchically from regional maps. PLAN's expertise, however, appears to be in maps whose scale and level of difficulty are far below those Ariadne traverses with ease, and the authors offer no statistics on either its efficiency or effectiveness.

A second way to characterize an approach to learning navigation for a robot is by its representation of its world. Hayes has constructed a rigorous approach to reasoning about space (Hayes 1988). He too envisions representations of space into pieces (like Ariadne's bottles, chambers, and quadrants) that have boundaries and connectors (like Ariadne's bottlenecks, access points, and gates). There is, as yet, no implemented version, however. TOUR (Kuipers 1978) and Qualnav (Kuipers & Levitt 1988) have landmarks that are sensorily distinctive. Ariadne's grid world, as originally postulated by Korf, did not provide such landmarks. As a substitute, Ariadne has gates and bases predicated upon theories about what might be useful in travel. Ariadne does not store routes or route fragments from completed problems either, whereas TOUR and PLAN both keep a topological network of routes between places that can be manipulated to find a path. Unlike these systems, Ariadne's global overview is implicit in the useful knowledge it learns; it plans naively and opportunistically. Although it would be simple enough to learn a route-fragment graph on Ariadne's gates and bases, the reactive approach described here is preferred, for both its computational efficiency and its limited storage.

8. Conclusions

Human navigators react quickly and correctly to clearly productive or unhelpful opportunities, such as "there's the goal" or "not that dead-end again." They entertain a variety of heuristics, such as "closer is better" or "when far away, take a long, straight step." They also digress into a search mode tailored for a particular situation, for example, "this obstacle is between me and the goal, so I have to get around it."

Ariadne is an implemented pragmatic navigation system that epitomizes this kind of naive geographic reasoning, as if it were learning the way around a new campus or town. Ariadne learns a variety of features about a new environment, and uses that knowledge to solve multiple travel problems there. Ariadne solves multiple problems in the same territory. It solves the easier ones in fewer steps and with fewer resources than more difficult problems. Compared to traditional search techniques, it solves these problems quickly, as measured in elapsed problem-solving

time, number of decisions, and path length. It also performs efficiently, as measured by number of distinct locations visited and the percentage of repeated locations in a path. And Ariadne learns, so that it applies previous experience to hitherto unseen problems, both in random mazes and in a variety of more realistic world models.

Facilitators and obstructers prove themselves to be a flexible and pragmatic foundation for pragmatic navigation. As the data indicate, learning is essential to an agent facing hard problems with limited resources. The ablated No-Learn agent failed to solve many of the hardest problems, while the full version could perform quite well after only 20 learning trials. The same useful knowledge can be applied (e.g., dead-ends) and learned (e.g., barriers) in many ways. Facilitators and obstructers appear to capture key navigational concepts about two-dimensional space, concepts that support robust and effective travel there.

Acknowledgments

Thanks to David Sullivan, Barry Schiffman, and Mike Pazzani for their thoughts on this effort. This work was supported in part by NSF grants #9001936 and #9423085, and PSC-CUNY grant #665292.

References

Branting, L. K. and Aha, D. W. (1995). Stratified Case-Based Reasoning: Reusing Hierarchical Problem Solving Episodes. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 384-390. Montreal: Morgan Kaufmann.

Chown, E., Kaplan, S. and Kortenkamp, D. 1995. Prototypes, Location , and Associative Networks (PLAN): Towards a Unified Theory of Cognitive Mapping. *Cognitive Science*, 19 : 1-51.

Egenhofer, M. J. and Mark, D. M. 1995. Naive Geography, 95-8, National Center for Geographic Information and Analysis.

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. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 454-461. Montreal: Morgan Kaufmann.

Ericsson, K. A. and Charness, N. 1994. Expert Performance: Its Structure and Acquisition. *American Psychologist*, 49 (8): 725-747.

Ericsson, K. A., Krampe, R. T. and Tesch-Römer, C. 1993. The Role of Deliberate Practice in the Acquisition of Expert Performance. *Psychological Review*, 100 (3): 363-406.

Ericsson, K. A. and Smith, J. (1991). *Toward a General Theory of Expertise - Prospects and Limits*. Cambridge: Cambridge University Press,

Golledge, R. G. (1995). Path Selection and Route Preference in Human Navigation: A Progress Report. In *Proceedings of the International Conference on Spatial Information and Theory (COSIT 95)*, 207-222. Semmerling, Austria: Springer Verlag.

Gryl, A. (1995). *Analyse et Modélisation des Processus Discursifs Mis en Oeuvre dans la Description d'Itinéraires*. Ph.D., Université Paris Xi Orsay.

Hayes, P. J. (1988). The Second Naive Physics Manifesto. In J. R. Hobbs, & R. C. Moore (Ed.), *Formal Theories of the Commonsense World* (pp. 1-36). Norwood, NJ: Ablex Publishing.

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

Kuipers, B. J. 1978. Modeling Spatial Knowledge. *Cognitive Science*, 2 : 129-153.

Kuipers, B. J. and Levitt, T. S. 1988. Navigation and Mapping in Large-Scale Space. *AI Magazine*, 9 (2): 25-43.

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

Piaget, J. and Inhelder, B. 1967. *The Child's Conception of Space*. New York: W. W. Norton.

Sadalla, E. K. and Magel, S. G. 1980. The Perception of Traversed Distance. *Environment and Behavior*, 12 : 65-79.

Sutton, R. S. (1990). 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: Morgan Kaufmann.

Tversky, B. 1991. Spatial Mental Models. *The Psychology of Learning and Motivation*, 27 : 109-145.

Tversky, B. 1992. Distortions in Cognitive Maps. *Geoforum*, 23 (2): 131-138.