This paper appeared in COSIT '97, pp.373-388.

Spatial Representation for Pragmatic Navigation

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

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.

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. 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, that is, useful approximations that describe a territory and travel experience through it. Such representation deliberately sacrifices detail in exchange for efficient storage, retrieval, and computation.

This paper describes the representation and reasoning that underlie *Ariadne*, a program for pragmatic navigation. (Ariadne, daughter of King Minos of Crete, told Theseus how to find his way through the labyrinth that protected a great treasure.) Instead of a map, Ariadne learns two kinds of features to describe a territory: those that support efficient travel (*facilitators*) and those make it more difficult (*obstructers*). The first section describes these features, and the second describes how that representation is harnessed to reason about travel through grid worlds. Subsequent sections analyze the source of the program's power, consider its cognitive plausibility, and discuss related work.

1. Learning to Represent Territory

Ariadne's world is described as 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 mazes substantially larger 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.

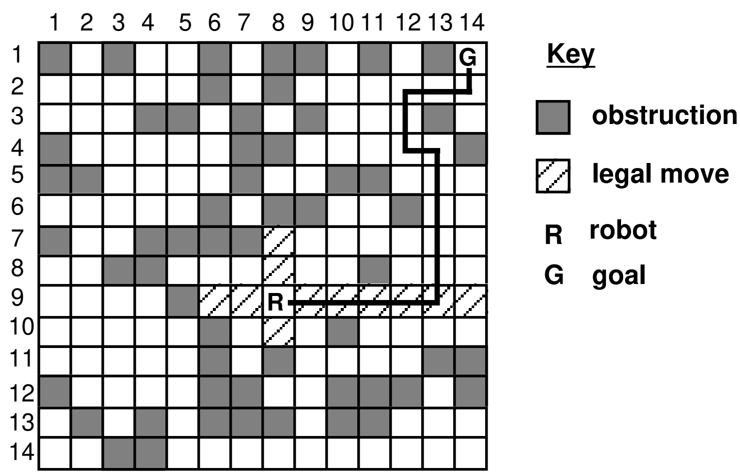


Fig. 1. A problem and its solution in a maze.

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. A problem is *solvable* if and only if there exists some path

$$\langle R = \mathrm{loc}_0\ \mathrm{move}_1\ \mathrm{loc}_1 \ldots \mathrm{loc}_{i-1}\ \mathrm{move}_i\ \mathrm{loc}_i \ldots \mathrm{loc}_{p-1}\ \mathrm{move}_p\ \mathrm{loc}_p = G \rangle$$

such that move_i is a legal move from loc_{i-1} to loc_i for $1 \leq i \leq p$. The *level of difficulty* of a solvable problem is the minimum value of *p* for which there is a solution, that is, one more than the minimum number of turns required 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. 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 quickly. Easier problems should be easier to solve. 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.

1.1 Facilitators

Ariadne identifies three kinds of facilitators: gates, bases, and corners. A gate may 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.

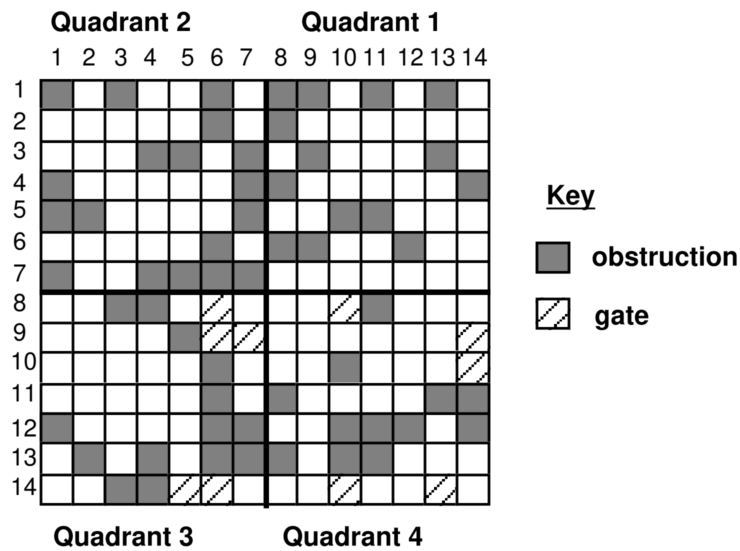


Fig. 2. After 10 tasks, gates learned for a simple maze.

Since it knows the dimensions of the maze, Ariadne can calculate quadrants. 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, (8, 10) is a gate between quadrants 3 and 4 in Figure 2, 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, for example, is the rectangle with vertices (6, 10), (9, 10), (9, 5), and (6, 5). (Observe how an extent may overestimate the set of actual such locations.) Ariadne only learns the gates it visits, so many locations in Figure 2 that satisfy the definition of gate were not learned and are not marked.

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; it was no different architecturally from most diners, 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 task; 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 solution fragments constructed by search that circumvents walls contribute only their 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.

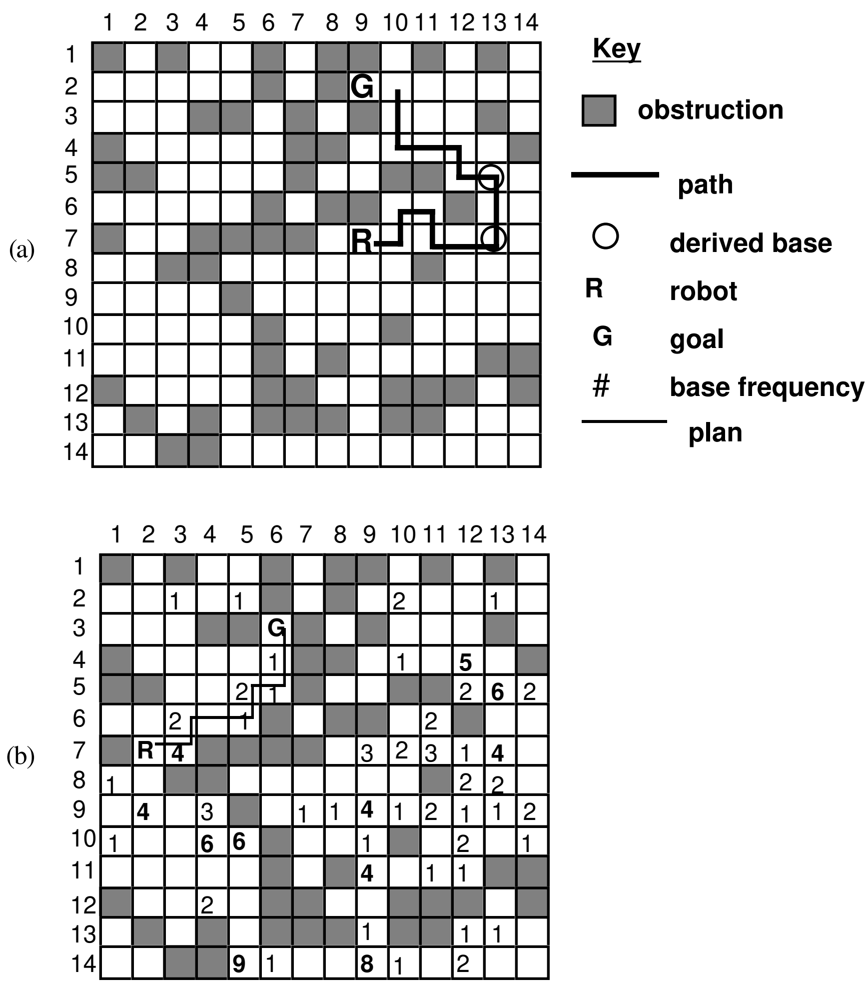


Fig. 3. (a) A solution path and the bases that arise from it. (b) A plan for a task formulated from learned bases.

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 task is shown in Figure 3(b), where the bases learned after 20 level-6 tasks 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.

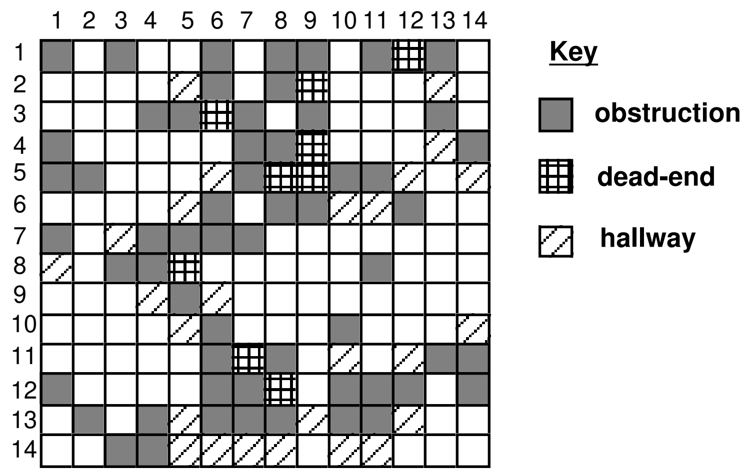


Fig. 4. After 10 tasks, corridors learned for a simple maze.

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 4, 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 4 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 4 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.

1.2 Obstructers

Ariadne identifies four kinds of obstructers: certain corridors as well as chambers, bottles, and barriers. A corridor may obstruct movement either because it is a dead-end, such as (2, 9) in Figure 4, 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 triggered when the robot has been *constrained* (hampered in its ability to move through space because it has recently covered little area or few distinct locations, or is moving repetitively) or has few novel choices.

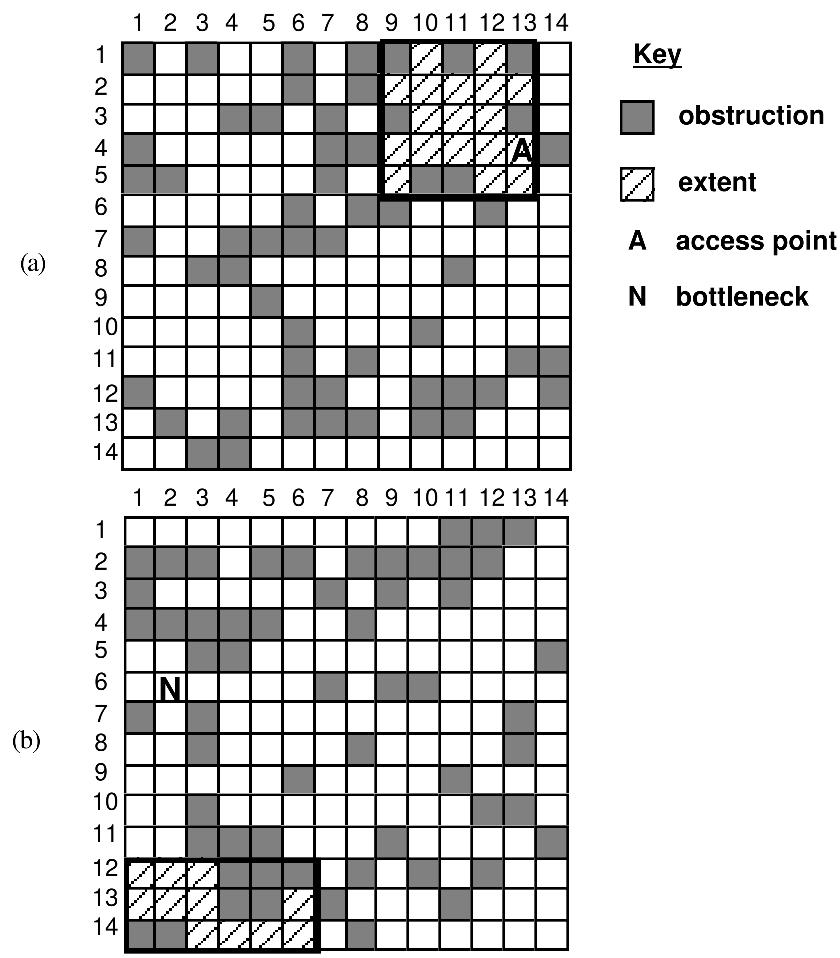


Fig. 5. (a) A learned chamber, with its extent and access point (b) A learned bottle with its neck and extent.

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 5(a) shows a chamber with extent 1 north, 13 east, 5 south, and 9 west. 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 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. If the procedure identifies a sequence of one or two locations that enlarge the extent, at least one of which is previously unvisited during this task, it records the chamber's extent and access point 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.

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 task 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 $x\%$ of the area of the maze. A bottle is identified as an extent and a *neck* (entry and/or exit point) is identified. A bottle's neck need not be contained within the bottle's extent, unlike the access point of a chamber. Bottles are stored in a hash table as an extent and a neck. Figure 5(b) shows a bottle with extent 12 north, 6 east, 14 south, and 1 west, and neck (6, 2).

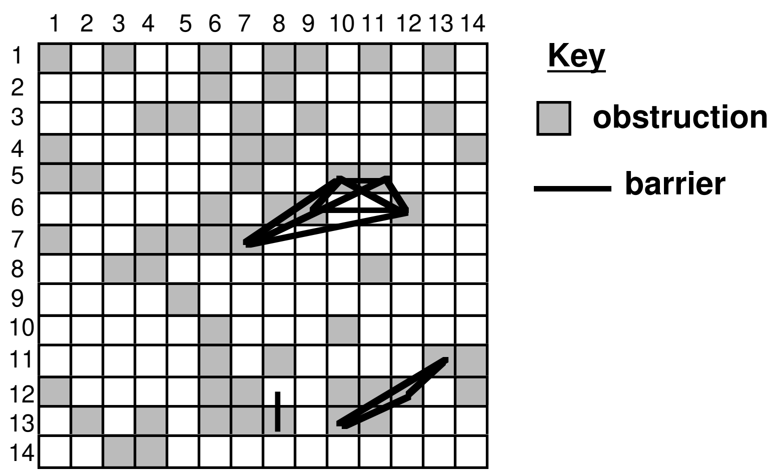


Fig. 6. After 10 tasks, barriers computed for a simple maze. Only some are retained.

A *barrier* is a linear approximation of a wall that obstructs movement. Figure 6 shows the barriers learned after 10 tasks in a simple maze. The barrier from (13, 10) to (11, 13), for example, is an approximation of the irregular wall in the lower right corner of the maze. Four decision making procedures have clearly prioritized search path preferences for the direction in which they move. Whether or not these procedures produce solution fragments, barriers are detected from the paths their searches followed. In each case, the learning algorithm is a function of the preferences in the rationale that produced the search path. Experience and Ariadne's recourse to search determine which barriers are encountered. Each barrier is an object with endpoints, slope, intercept, and length.

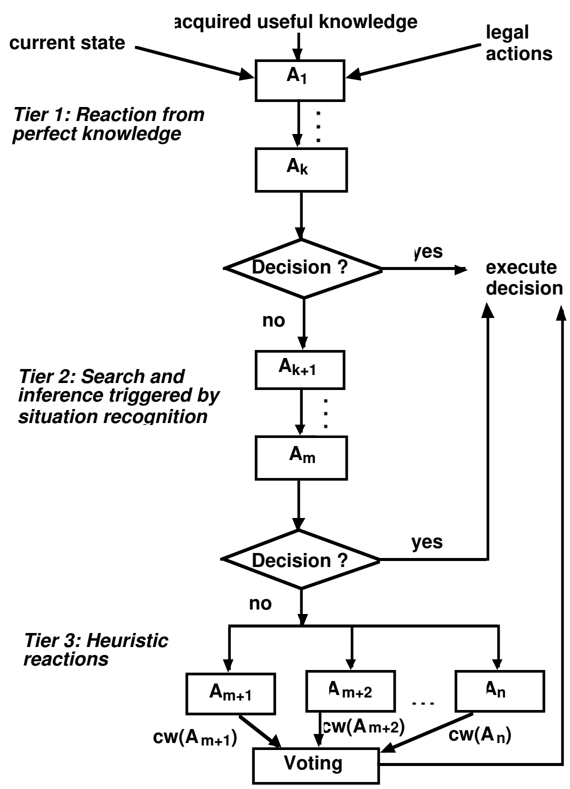


Fig. 7. How FORR makes decisions.

2. Reasoning with the Representation

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 [4]. 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.

FORR's three-tier hierarchical model of the reasoning process (shown in Figure 7) 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 [5]. A full listing of Ariadne's 32 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 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 procedures that do time-limited search in an attempt to produce a sequence of moves that they then mandate. Each 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 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 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. Crook advocates a move to the near end of a corridor that is neither straight nor a dead-end.

Some Advisors simply avoid obstructers. No Way keeps the robot out of dead-ends that could not contain the goal, and Pipeline does the same for pipes. Outta Here, Roundabout, and Wander all avoid dead-ends during their 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 searches for an exit from a chamber if the goal is not there, and an entrance to it if the goal might be there. Chamberlain is a heuristic, non-search-based version of Outta Here, and Cork does for bottles what Chamberlain does for chambers. Probe learns chambers as a side effect of limited search.

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

Tier	Advisor	Rationale	Useful knowledge
1	*Victory*	Go to goal reachable by one legal move.	—
1	*Can Do*	Move adjacent to goal.	—
1	*No Way*	Avoid dead-ends.	Corridors
1	*Pipeline*	Avoid internal locations in pipes.	Corridors
2	*Roundabout*	Circumnavigate obstructions.	Barriers, corridors
2	*Outta Here*	Exit chamber or dead-end not containing goal.	Chambers, corridors
2	*Probe*	Try to leave current extent.	—
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 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 bottle that contains goal; avoid bottle that does not.	Bottles
3	*Corner*	Move to end of pipe that promises turn.	Corridors
3	*Crook*	Move to end of crooked corridor.	Corridors
3	*Cycle Breaker*	Stop repeated visits to same few spots.	—
3	*Detour*	Move away from barriers obstructing goal.	Barriers
3	*Done That*	Discourage moves in same direction as before from a previously-visited location.	—
3	*Giant Step*	If recently confined, take 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 steps
3	*Leap Frog*	Execute opportunistic plans.	Bases
3	*Mr. Rogers*	Move into neighborhood of goal.	—
3	*Opening*	Begin as previously successful path did.	Openings
3	*Plod*	Take 1-unit step, preferably toward goal.	—
3	*Quadro*	Move into goal's quadrant or into new quadrant.	Gates

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.

3. Empirical Design and Results

The data described here was produced when Ariadne generated a maze and tested the performance there of different reasoning agents. Because FORR is non-deterministic, results from 10 runs (i.e., 10 different randomly-generated mazes) were averaged to produce an *experiment*. Throughout this section, cited differences are at the 95% confidence level unless otherwise stated.

3.1 The ablation experiments

The first set of experiments demonstrates Ariadne's ability after learning to solve problems it has never before experienced, and explores which components of the program are necessary to achieve such performance. These experiments were performed on *random mazes,* that is, 20×20 mazes that were 30% obstructed, with problem levels 4, 6, 8, and 10. In each maze, the full version of Ariadne was given 20 learning problems (*R* and *G* pairs) at a fixed level of difficulty. Then 10 newly-generated testing problems for the same maze and level of difficulty were offered. A learning or testing problem was terminated when the agent reached the goal or when it reached the decision step limit. On levels 4 and 6, this limit was 200; at higher levels it was 1000.

After learning, the following agents were tested: Ariadne, the *Reactive agent* that used only the tier-1 Advisors, the *Reactive+Search agent* that used only the tier-1 and tier-2 Advisors, the *Reactive+Heuristic agent* that used only the tier-1 and tier-3 Advisors, the *No-Learn agent* that applied all the Advisors but made no useful knowledge available, and the *No-Plan agent* that applied all the useful knowledge and all the Advisors except those involved with planning (Leap Frog, Home Run, and Patchwork). (A separate experiment tested a *random agent* that selected random legal moves. Even on level 4, the random agent was only able to solve so few problems, and those with such lengthy paths, that it was eliminated from further testing.)

An agent was eliminated from testing on all subsequent levels if it was significantly slower than Ariadne, traveled significantly further, or failed to solve half the problems within the decision-step limit at any given level. Table 2 reports the results, where 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.) The Reactive agent and the Reactive+Search agent were eliminated after level 4, the Reactive+Heuristic agent after level 6, and the No-Learn agent after level 8. On level 10, Ariadne is shown superior to the No-Plan agent in both path length and computation time.

Table 2: The performance of Ariadne and ablated versions of it after learning in a particular 20×20 maze in Ariadne's world. Results are averaged over runs in 10 mazes. "Distance" is the Manhattan distance along the path to the goal, "decisions" the number of steps taken during tier-2 search or solution, "moves" the number of steps in the solution, "locations" the number of distinct locations actually visited, and "time" execution time per testing problem in seconds.

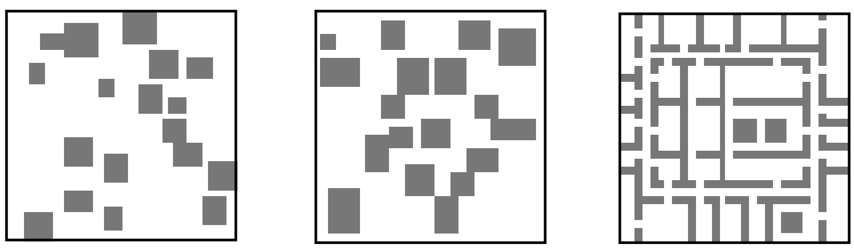
	Agent	Distance	Decisions	Moves	Locations	Time
4-step problems						
generator path length 13.56						
	Reactive	125.02	69.06	47.58	26.00	5.69
	Reactive+Search	32.64	49.07	16.15	13.88	1.64
	Reactive+Heuristic	29.71	13.15	11.25	9.53	2.44
	No-Learn	23.77	35.27	11.99	10.17	1.41
	No-Plan	15.87	23.56	10.33	9.88	1.13
100%	Ariadne	18.74	20.33	11.64	10.95	1.21
6-step problems						
generator path length 20.54						
	Reactive+Heuristic	46.04	20.67	16.99	13.80	3.61
	No-Learn	40.33	58.53	21.77	18.02	2.38
	No-Plan	27.60	32.77	16.64	15.93	1.84
93%	Ariadne	27.23	47.61	16.22	15.15	1.99
8-step problems						
generator path length 24.83						
	No-Learn	99.08	202.85	44.10	28.39	7.08
	No-Plan	40.81	76.89	26.06	23.58	3.71
95%	Ariadne	38.15	115.19	24.16	22.19	4.00
10-step problems						
generator path length 31.46						
	No-Plan	80.02	151.87	40.33	34.44	9.96
98%	Ariadne	61.29	110.56	34.70	30.08	6.63

The problem generator produces a problem whose difficulty is predicated on number of required turns rather than path length. In reality, however, it is often possible to find a shorter solution with more turns, and Ariadne frequently does so. On level 4 during testing, Ariadne found as short or shorter a path as that of the problem generator 42% of the time, 40% on level 6, 26% on level 8, and 16% on level 10. Inspection indicates that most of the rest of Ariadne's solutions are near optimal, with an occasional outlier or two driving up the average distance.

To measure the efficacy of Ariadne's problem solving, compare it with two standard AI techniques: breadth-first search and heuristic *best-first search* with an evaluation equal to the Euclidean distance from the robot to the goal. On level 10, Ariadne only visits 10.74% of the locations accessible to the robot from its starting position. In contrast, breadth-first search solved the same problems while exploring 88.96% of the maze. This somewhat understates the cost of a physically executed breadth-first search, whose many repetitive subpaths go uncounted here. In another experiment of 10 runs where Ariadne first learned on 20 problems, best-first search with only the Euclidean distance to the goal as its evaluation function was able to solve only 37% on level 10 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.

3.2 Performance in non-random environments

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 8. 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 with outside, corner suites accessible only through an adjacent office.



(a) (b) (c)

Fig. 8. Some non-random environments: (a) warehouse (b) furnished room (c) office. Grids are omitted for clarity

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 8. In each problem class on each run a new maze (e.g., a warehouse or a furnished room) was generated. Ariadne learned on 20 different problems in that maze, and then was tested on 10 previously-unseen problems in the same maze. 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, presumably 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 the time as well or better than the problem generator's solution. 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 10% 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 deceiving to demand additional search. This appears to be evidence that Ariadne's knowledge representation is adequate for most two-dimensional grid worlds, and that its satisficing approach scales.

4. Discussion

Ariadne's pragmatic approach is effective on a task not amenable to traditional AI techniques. The long and repetitive paths of depth-first search requires substantial backtracking. 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. (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.

There is no claim here that Ariadne is a cognitive model of a human navigator, only that many of its features are cognitively plausible. Bases are reminiscent of landmarks, although for their spatial, rather than their visual, properties. Several of the Advisors do model principles of naive geographic reasoning [3] 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 [9] and the rationales of several of the other Advisors [10]. 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 [6; 7; 8; 14]. Such expertise is applicable to a related set of problem classes. For example, an expert path finder in unfamiliar territory will 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 [17]. These representations systematically distort visual perception to facilitate recall [18]. 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 [3]. Even the use of levels (number of turns) for degree of problem difficulty is supported by results that people gauge distance that way [15].

The spatial representation postulated here supports performance in the more realistic environments of Section 3.2, and is readily adapted further. Winding streets, for example, could be plotted using a finer level of detail in the grid, and an Advisor that continues along a street could be included to compensate for the changes in street direction. Still, there is no claim that the representation is complete. If altitude were significant to path finding in an environment, some revision would certainly be required. In addition, Ariadne's occasional difficulties with the deliberately constructed corner suites in office mazes suggest that some form of chaining may also be necessary, either as a new representation or as a way to link existing representations together. Ariadne does demonstrate, however, how expertise can be developed rapidly by a navigator with limited sensors.

Other learning methods have been applied only to mazes far easier than those Ariadne works in: temporal difference [16], case-based planning [1], and a connectionist model that integrates path finding into general cognition [2]. Hayes has constructed a rigorous approach to reasoning about space, as yet unimplemented [11]. TOUR [12] and Qualnav [13] have landmarks that are sensorily distinctive, features that this work has yet to incorporate.

Ariadne is an implemented pragmatic navigation system that epitomizes 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 and efficiently. 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. Trials on more realistic grid worlds suggest that they will scale well. This representation appears to capture key navigation concepts about two-dimensional space, concepts that support robust and effective travel there.

References

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

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

[3] M.J. Egenhofer and D.M. Mark. *Naive Geography* (95-8). National Center for Geographic Information and Analysis. 1995.

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

[5] S.L. Epstein. On Heuristic Reasoning, Reactivity, and Search. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 454-461, Montreal, 1995. Morgan Kaufmann.

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

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

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

[9] R. G. Golledge. 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, 1995. Springer Verlag.

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

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

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

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

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

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

[16] 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.

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

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