Combining Search and Analogical Reasoning in Path Planning from Road Maps *

Karen Haigh **Manuela Veloso**
School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213
{khaigh,mmv}@cs.cmu.edu

Abstract

Path planning from road maps is a task that may involve multiple goal interactions and multiple ways of achieving a goal. This problem is recognized as a difficult problem solving task. In this domain it is particularly interesting to explore learning techniques that can improve the problem solver's efficiency both at plan generation and plan execution. We want to study the problem from two particular novel angles: that of real execution in an autonomous vehicle (instead of simulated execution); and that of interspersing execution and replanning as an additional learning experience. This paper presents the initial work towards this goal, namely the integration of analogical reasoning with problem solving when applied to the domain of path planning from large real maps. We show how the complexity of path planning is related to multiple ways of achieving the goals. We review the case representation and describe how these cases are reused in path planning where we interleave a breadth-first problem solving search technique with analogical case replay. Finally, we show empirical results using a real road map.

Introduction

The motivation and long-term goal of this work is to integrate planning, real execution, and learning by analogy in the domain of traversing road maps to achieve multiple goals. PRODIGY, a planner and learner [Carbonell *et al.*, 1990b], will be integrated with an autonomous navigation vehicle which will execute the plans to achieve multiple goals while driving in a city. The planner will be given a road map, a set of goals, and the initial location of the vehicle. It will generate a near-optimal route that achieves all the goals. We intend to use NAVLAB [Thorpe, 1990], an autonomous vehicle driven by a set of neural networks to execute the plan. NAVLAB will combine low-level perception with the high-level reasoning of the plan which will guide it in making more complex decisions such as which way to turn at intersections in order to achieve the goals. Real execution of the plan may lead to failures of planned steps, such as a blocked road. The vehicle will transmit this information to the planner for replanning and learning.

Several researchers investigate the problem of interleaving planning and execution [Hammond *et al.*, 1990, Agre and Chapman, 1987, McDermott, 1978]. In this work we want to study the problem from two particular angles: that of real execution in an autonomous vehicle (instead of simulated execution), and that of interspersing execution and replanning as an additional learning experience. We envision breaking the representation gap between a high level reasoning planner and a vehicle in the real-world executing the plan.

We report on preliminary work towards reaching this motivating scenario. This paper focuses on the development of a robust planning and learning system where we accumulate a library of cases as planning episodes to guide the initial planning as well as any replanning needed at execution time.

Path planning for multiple goals involves a large search space with a large set of alternative ways to achieve each individual goal and many possible goal interactions. We initially investigate the issues of designing the domain using real road maps. Then we discuss how analogical reasoning applies to problem solving in this domain and show empirical results on the integration with depth-first and breadth-first search and discuss our on going implementation of best-first search. We explore in particular how case reuse affects

*This research was sponsored by the Avionics Laboratory, Wright Research and Development Center, Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, Arpa Order No. 7597. The first author is also supported in part by the Natural Sciences and Engineering Research Council of Canada. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government or of the Canadian Government.

the planning time, since it is very important to reduce the search space, especially when replanning during execution.

Path Planning from Road Maps

Introduction

The problem examined in this paper is how to find a path in a map when there are multiple goals. Goals may consist of moving to different locations, getting orders, and/or delivering packages.

Path planning in graphs has been addressed by a variety of algorithms, such as Dijkstra's shortest path algorithm [Aho *et al.*, 1974]. However, since our goal is to implement this in autonomous vehicles, we want to be able to interleave path planning with execution. Paths will have to be modified and altered during driving (because of detours, for example) and, therefore, we need a method which is more flexible than a shortest path algorithm, and where we can reuse previous experience as in [Goel *et al.*, 1992].

Furthermore, our framework with real road maps diverges from the more general framework of path planning in arbitrary graphs. In fact, real road maps are not static since they will have minor temporary variations which will only be known at execution time. The path planning problem is also characterized by an extremely large number of alternative ways of reaching target destinations, many of which will be equivalent from a distance point of view. The emphasis of this work is therefore on learning from experience in a real environment rather than a simulated one. We use analogical reasoning to enable the planner to accumulate and reuse its planning experience.

Finally, our path planning process is not driven exclusively by finding the path with the minimum distance between locations. Our aim is rather to find acceptable solutions to multiple goals which can be achieved in several alternative ways, a problem reducible to the Hamiltonian circuit and therefore NP-hard. Reusing previous experiences will reduce this complexity.

The Domain Representation

When designing a domain representation, we considered how the representation would affect the ultimate goal: using this planner in the autonomous vehicle. We need a representation that adequately describes one way streets, distances between the initial position and the destination, and the direction of turns that will need to be made.

Undirected graphs do not suffice since one way streets can not be represented, and although directed graphs would be able to handle this problem, there is no easy way to represent turning direction. We are therefore using a system which explicitly connects one city block with the next one, thereby allowing us to store all this required information. The streets are therefore divided into multiple segments separated by intersections. We recently found access to a detailed database of the complete Pittsburgh street map with more than 20,000 street segments. The database includes the spatial coordinates of the intersections which we will use to define the turning direction between street segments [Bruegge *et al.*, 1992].

As an example, in our current simplified representation, the predicate `(connected Street1 Street2 Distance Turning-direction)` encodes state information about the map. The domain is encoded as a series of operators that describe possible actions. Currently we have operators that move agents out of and into buildings, and move agents between streets. For example, the (goto-adjacent-street <street1> <street2> <distance>) operator moves the agent between two adjacent streets, namely from <street1> to <street2>. The preconditions of this operator require that the two streets be connected, and that the agent is at <street1>. It also calculates the total distance travelled by adding the new distance to the current total.

Standard Means-End Analysis Search

Because of the reasons described in the introduction of previous section, we used the PRODIGY planning and learning system [Carbonell *et al.*, 1990a]. PRODIGY's nonlinear planner uses means-ends analysis in its backward-chaining search procedure which can reason about multiple goals and multiple alternative operators relevant to the goals. This choice of operators amounts to multiple ways of trying to achieve the same goal.

The planning reasoning cycle involves several decision points, namely: which *goal* to select from the set of pending goals and subgoals; which *operator* to choose to achieve a particular goal; which *bindings* to choose to instantiate the chosen operator; and whether to *apply* an operator whose preconditions are satisfied or to continue *subgoaling* on a yet unachieved goal.

Dynamic goal selection from the set of pending goals enables the planner to interleave plans, exploiting common subgoals and addressing issues of resource contention. The planner returns a partially ordered plan as a result of analyzing the dependencies among the steps in the totally ordered solution found while planning.

In a typical road map, each street-section is connected to at most eight other street-sections. This branching factor varies between two as the lowest value (i.e. a dead-end street) and eight as the highest (five-way intersections at both ends). The branching factor in the map used for our experiments ranges between two and six, and averages about 4.4.

Therefore for a problem of travelling n city blocks, search complexity is loosely bounded above by approximately 8^n. This number is reduced by the fact that the average branching factor is lower, and also by elim-

n	$\sum_{i=0}^{n} 4.4^i$	$\sum_{i=0}^{n} 8^i$
5	2134	3745
10	3.52×10^6	1.23×10^9
15	5.80×10^9	4.02×10^{13}

Table 1: Complexity of search space (in number of nodes expanded) for travelling n city blocks

inating goal loops, for example <goto-adjacent-street x y > followed by <goto-adjacent-street y x > [Carbonell *et al.*, 1992]. For even relatively small values of n, however, reaching a solution by straight-forward breadth-first search is an extremely slow and tedious process (see Table 1).

Case Reuse Combined with Search

Given the complexity described above, we feel that applying analogy and case-based reasoning in the context of map path planning is highly appropriate and even necessary given the time restraints required when interleaving planning with execution. We can reuse cases in this context because one solution path will often be a subpath of another problem. If the smaller problem has already been solved, we can then reuse it and significantly reduce the amount of search necessary to find the new solution.

Case Representation

In the original solution path, PRODIGY had to make various decisions about which paths to follow. Of all the nodes generated while solving a problem, only the ones on *the solution path* are stored to create a case. Extraneous nodes are discarded. Each *relevant* decision from the original solution (i.e. each node in the solution path) and its justification is stored in LISP format in order to make reloading the case in a PRODIGY-readable format very easy. Figure 1 contains an example of how a goal node would be stored. Essentially, it maintains all pointers to related nodes in the search tree (which operators introduced it, any other applicable operators left, remaining goals). A complete case contains nodes of a similar format describing decisions and justifications for decisions made at operator nodes, binding nodes and applied operator nodes.

```
(setf (p4::nexus-children (find-node 4))
    (list (p4::make-goal-node
            :name 5
            :parent (find-node 4)
            :goal (p4::instantiate-consed-literal
                        '(AT BARTLETT-2 JANE ))
            :introducing-operators
                        (list (find-node 4) ))))
```

Figure 1: A sketch of a goal node in a case.

Note that a case is not used as a simple "macro-operator" [Fikes and Nilsson, 1971]. A case is selected based on a partial match to a new problem solving situation. Hence, as opposed to a macro-operator, a case *guides* and *does not dictate* the reconstruction process. In addition, intermediate decisions corresponding to choices internal to each case can be bypassed or adapted if their justifications no longer hold.

Case Reusage

We follow the case reuse strategy as developed in [Veloso, 1992]. The replay technique involves a closely coupled interaction between planning using the domain theory, domain operators, and similar cases which are derivational traces of both successful and failed decisions in past planning episodes. The replay mechanism involves a reinterpretation of the decision justifications in the context of the new problem, reusing past decisions when the justifications hold true, and replanning using the domain theory when the transfer fails.

Once one (or more) case is is found that is similar to the new problem solving situation, it is ready to be reused. The planner is called and given the the set of operators and the similar case as input. The replay algorithm is implemented by interrupting the planning algorithm at its decision points so that it may make choices similar to the ones from the guiding case.

Until there is a match between a subgoal of the case and one of the candidate goals of the new problem, PRODIGY does breadth-first-search to maximize the chance that a match will be found with minimum depth. As soon as this match has been found, PRODIGY immediately follows the case using depth-first-search, and does not expand the rest of the nodes at the same level as the matched node. Once case nodes for which similarity justifications hold have been exhausted, PRODIGY returns to breadth-first-search until its main goal state has been achieved (see Figure 2). This method allows us not only to minimize additional searching, but also to solve problems in which neither the goal state nor the initial state are the same as the original case.

Note that, optimality of paths is not necessarily preserved by analogical transfer. The merging of optimal subplans under a satisficing approach may result in a non-optimal new plan. When there are multiple operators to achieve goals, there is no known technique that both tries to maximize the reuse of previous experience and also maximize the quality of the new similar solution. We plan to investigate an exploration technique that allows untried or unjustified steps in the new context to be searched, diverging from the direct reuse of the past experience. This exploratory search can be conducted when the planner is not otherwise occupied. It should be noted however, that we are not explicitly concerned with always finding an optimal solution, but rather with finding a reasonable solution.

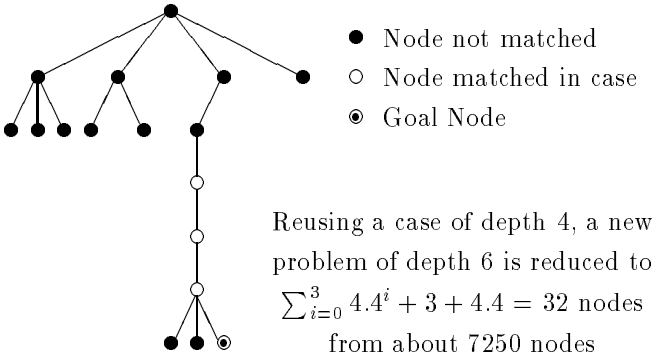

Figure 2: PRODIGY's node expansion tree when problem solving using a case

Experiments using a Real Map

The Map

The map we used is that of Pittsburgh's Squirrel Hill district, adjacent to Carnegie Mellon University and typical of a city environment in which the autonomous vehicle might be used. Besides the residential housing, there is a small shopping strip in the area, a university and a golf course, giving the vehicle a reasonably large region in which to run errands. The region contains two one-way streets (Darlington-5 and 6, and Bartlett-4 and 5) and several dead-end streets. This diversity, combined with the forty-eight intersections, causes the search space to be highly complex. Figure 3 shows a graphical version of the current representation of the map.

Results

The experiments we ran were constructed by extending a base case involving two operators. There were eight composable problems built around each base case: extending the base case by one operator (street) at the initial point, extending it by two operators at the initial point, and then each of those three cases were extended by one and two operators at the goal. Base cases were randomly chosen from the map of Squirrel Hill, and the composable problems were manually selected for their proximity to the base case.

We first ran each problem with a breadth-first search with no analogy until the solution was found, and then ran the same problem using depth-first search with the minimal depth-bound necessary to find a solution. Once these two runs were complete, we ran the same problem using analogy with all the cases that formed sub-paths to the solution.

We expected that the total number of nodes expanded would be reduced from $\sum_{i=0}^{d} 4.4^i$ where d = depth of the search, to a number Order of $\sum_{i=0}^{S} 4.4^i + C + \sum_{j=1}^{E} 4.4^j$ where S = the number of operators added at the start, C = number of nodes in the case's solution path, E = number of operators by which the case was extended at the end, and $S+C+(E-1) = d$.

Breadth-first search and depth-first search behaved as expected. Breadth-first search for a solution involving six operators required between 6,600 and 30,000 nodes, averaging about 15,000. The number of nodes expanded in a depth-first search ranged from finding the goal in the minimum number of nodes possible, to finding the goal with nearly the same number of nodes as the equivalent breadth-first search.

In all the problems solved by reusing cases, the number of nodes expanded was reduced as drastically as expected. Figure 4 shows a graph of our results, where '1S' (2S) represent problems built upon a case which required adding one (two) <goto-adjacent-street> operators at the Start and '1E' (2E) represent problems built upon a case which required adding one (two) operators at the End in order to reach a solution.

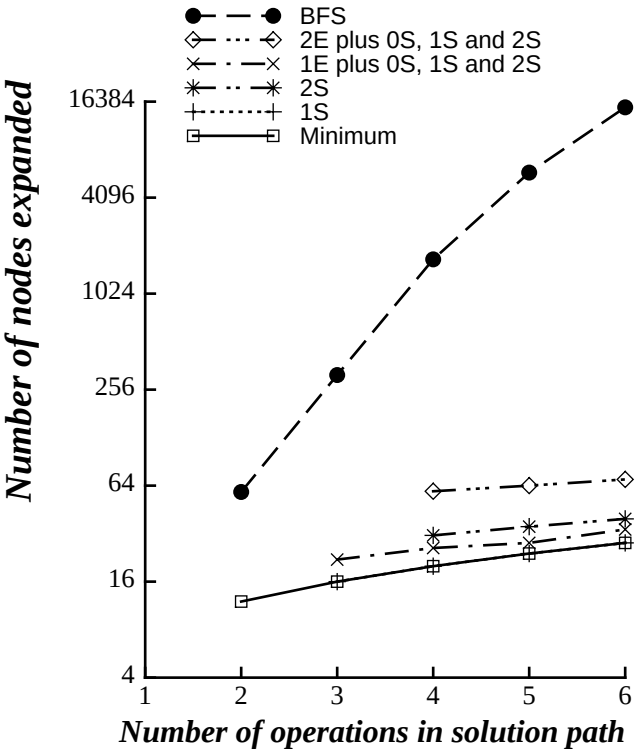

Figure 4: Problem Solving using BFS as compared to case reusage. Note that the numbers are four times greater than the number of operators because PRODIGY creates four nodes per operator.

The way PRODIGY applies operators in its means-ends analysis search accounts for the asymmetry between adding one (two) step at the beginning of a case and adding one (two) at the end. At the beginning of the search, PRODIGY will stop expanding nodes as soon as it finds a match. This situation explains why

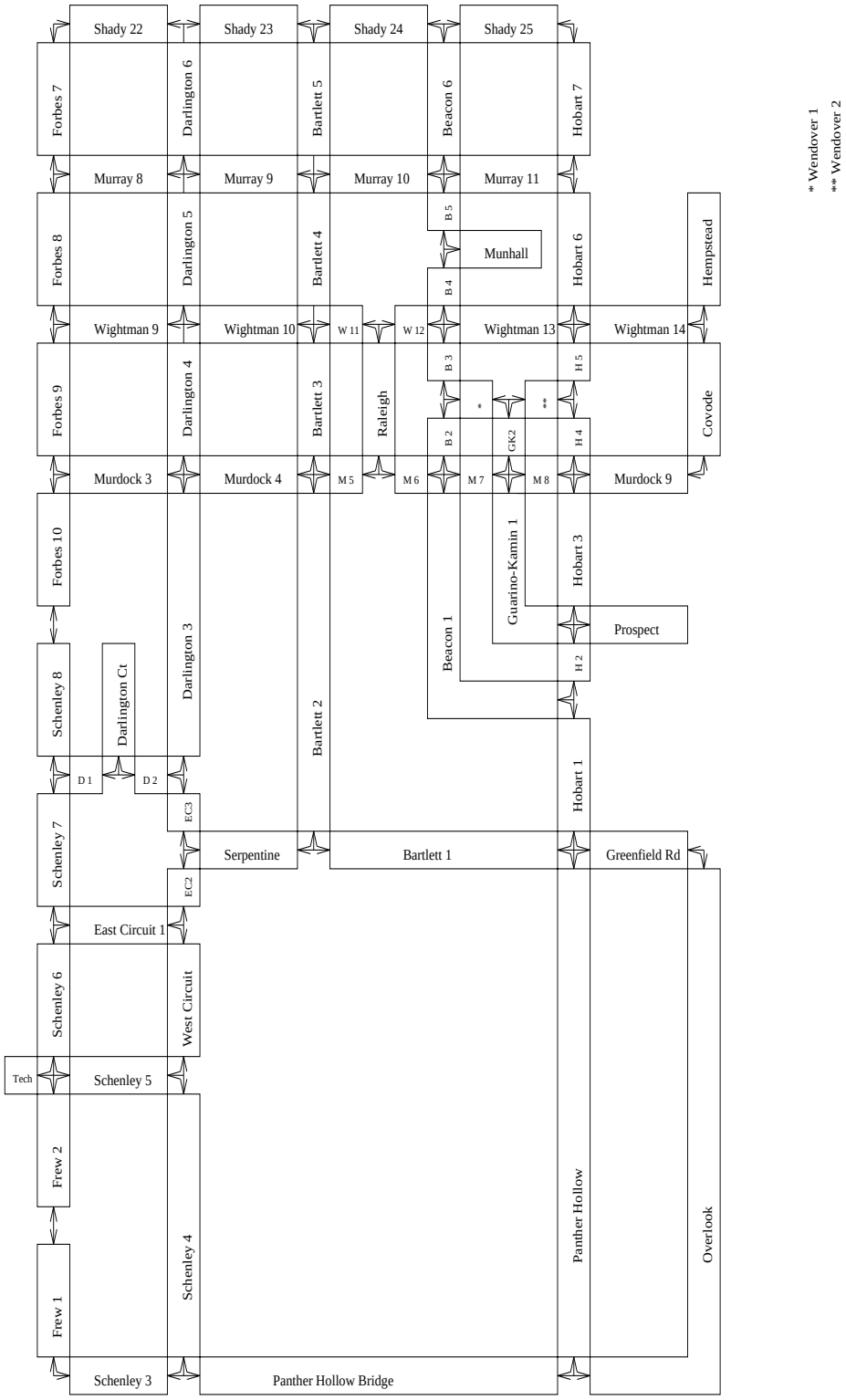

Figure 3: The Map: Pittsburgh's Squirrel Hill

'1S' cases so closely approximated the minimum number of nodes possible to find a solution. Once the case has been exhausted, PRODIGY continues with breadth-first-search. Even if it can apply an operator on one branch of the search, it may not yet have reached the final goal. Meanwhile, the other branches of the search tree will still have subgoals to expand, and thereby create more nodes at the end of the problem than at the start.

The number of nodes expanded at the end of the case so dominated the number of nodes expanded at the beginning that we combined all the '1E' cases and all the '2E' cases for the purposes of simplifying the graph.

We also ran a few experiments in which the case used did not form a proper sub-path of the optimal solution of the new problem, generating a solution that was non-optimal by one or two operators. The number of nodes expanded by PRODIGY was approximately double that of the proper sub-path problems, but even the most difficult were solved in less than one hundred nodes; several orders of magnitude less than the equivalent breadth-first-search.

Use of analogy instead of breadth-first-search resulted in a reduction in computation time from several hours to under a minute for longer problems. This fact indicates that this system will be usable in the real-time environment of interleaving planning and execution.

Notice that the representation used and the experiments run are of reduced complexity. In this initial phase we focused on developing and validating the basic framework. Refinements and extensions of the approach will result from the integration with the real autonomous vehicle.

Discussion and Future Work

The paper presents our initial accomplishments towards having a planner efficiently plan paths in a real road map. We implemented a real map of a considerably large part of Pittsburgh's Squirrel Hill district. Case reusage and analogical reasoning in path planning with road maps is compatible with human intuition since not only is the road map the same in each problem and planning situations similar, but finding solutions requires a lot of computation and search. We have shown in this paper that reusing cases in this context is feasible and efficient.

We are currently developing a large case library and organizing it for efficient retrieval [Doorenbos and Veloso, 1993]. We are using spatial features of the maps for case indexation. We are extending the cases in the library by planning with more operators, adding multiple goals, and generating alternative plans.

Secondly, since plan quality might not be preserved by analogical reasoning with extension and adaptations of problems, we plan to develop an exploratory mode within the system. This addition will allow us to not only ensure that optimal solutions are found for problems solved using analogy, but also to store multiple optimal solution paths for one problem.

In this domain we will also be investigating the merging of previous solutions so that problems can be solved using more than one case, either by directly linking several paths, or by merging subgoals [Veloso, 1993].

Introducing abstraction planning to this domain is also in our research agenda. We will extend the re-use and generalization of cases to different levels of abstraction, for example highway movement as opposed to major streets as opposed to minor streets. Another possible method of abstracting this kind of problem is by moving between grid squares (i.e. A-3 to J-6). Since abstraction has already been implemented within PRODIGY [Knoblock, 1991], we envision a smooth integration.

Finally, our immediate focus is to connect the planner to the autonomous vehicle and set up the appropriate communication framework.

Acknowledgements

The authors would like to thank Robert Driskill, Eugene Fink, and Bob Doorenbos for comments and suggestions on this paper as well as the whole PRODIGY research group for helpful discussions.

References

Agre, Phillip and Chapman, David 1987. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, San Mateo, CA. Morgan Kaufmann. 268–272.

Aho, A. V.; Hopcroft, J. E.; and Ullman, J. D. 1974. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, Massachusetts.

Bruegge, Bernd; Blythe, Jim; Jackson, Jeff; and Shufelt, Jeff 1992. Object-oriented system modeling with omt. In *Proceedings of the OOPSLA '92 Conference.* ACM Press. 359–376.

Carbonell, Jaime G.; Gil, Yolanda; Joseph, Robert; Knoblock, Craig A.; Minton, Steven; and Veloso, Manuela M. 1990a. Designing an integrated architecture: The PRODIGY view. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, MIT, MA. 997–1004.

Carbonell, Jaime G.; Knoblock, Craig A.; and Minton, Steven 1990b. Prodigy: An integrated architecture for planning and learning. In VanLehn, K., editor 1990b, *Architectures for Intelligence.* Erlbaum, Hillsdale, NJ. Also Technical Report CMU-CS-89-189.

Carbonell, Jaime G.; ; and PRODIGY Research Group, the 1992. PRODIGY4.0: The manual and tutorial. Technical Report CMU-CS-92-150, School of Computer Science, Carnegie Mellon University.

Doorenbos, Robert B. and Veloso, Manuela M. 1993. Knowledge organization and the utility problem. Technical Report forthcoming, School of Computer Science, Carnegie Mellon University.

Fikes, Richard E. and Nilsson, Nils J. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189–208.

Goel, Ashok; Donnelan, Michael; Vazquez, Nancy; and Callantine, Todd 1992. An integrated experience-based approach to navigational path planning for autonomous mobile robots. In *Working notes of the AAAI Fall Symposium on Applications of Artificial Intelligence to Real-World Autonomous Mobile Robots*, Cambridge, MA. 50–61.

Hammond, Kristian; Converse, Timothy; and Martin, Charles 1990. Integrating planning and acting in a case-based framework. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, San Mateo, CA. Morgan Kaufmann. 292–297.

Knoblock, Craig A. 1991. *Automatically Generating Abstractions for Problem Solving.* Ph.D. Dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. Available as technical report CMU-CS-91-120.

McDermott, Drew 1978. Planning and acting. *Cognitive Science* 2.

Thorpe, Charles E., editor 1990. *The CMU Navlab.* Kluwer Academic Publishers, Boston, MA.

Veloso, Manuela M. 1992. *Learning by Analogical Reasoning in General Problem Solving.* Ph.D. Dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. Available as technical report CMU-CS-92-174.

Veloso, Manuela M. 1993. Planning for complex tasks: Replay and merging of multiple simple plans. In *Preprints of the AAAI 1993 Spring Symposium Series, Workshop on Foundations of Automatic Planning: The Classical Approach and Beyond*, Stanford University, CA.