Computational Trade-offs in Experience-Based Reasoning

Ashok K. Goel and **Khaled S. Ali** and **Andrés Gómez de Silva Garza**
College of Computing, Georgia Institute of Technology
801 Atlantic Drive, Atlanta, GA 30332-0280
{goel,kali,andres}@cc.gatech.edu, (404)-853-9371

Abstract

Navigational path planning is a classical problem in robotics. Traditional approaches use goal-directed heuristic search of problem spaces defined by spatial models of the navigation world. Case-based reasoning offers an alternative approach. In the Router project, we have combined the case-based method with the model-based method. Since Router is a multistrategy system, it provides an experimental testbed to study some of the hypotheses of case-based reasoning. In this paper, we report on a set of experiments that examine four hypotheses: (i) the case-based method is more efficient than the model-based method, (ii) the case-based method produces plans of quality equal to those produced by the model-based method, (iii) the case-based method requires less knowledge but has the same problem-solving coverage as the model-based method, and (iv) cases need to be decomposed into partial cases for efficient and effective problem solving. We find that while hypothesis (i) is true, the others are questionable.

Goals and Motivations

Navigational path planning is a classical problem in autonomous mobile robotics. Qualitatively, a good method for robot planning would obey the following constraints: (1) since robots have access only to bounded computational resources, the planning method would require only limited processing and memory, (2) since robots need to perform in close to real time, it would (a) successfully form accurate plans and (b) form them very efficiently; and (3) since robots often operate in dynamic worlds, it would not assume complete and correct knowledge of the world. The contradictions between constraints 1 and 2, 2 and 3, 2(a) and 2(b), make navigational path planning a challenging problem for AI.

Most AI approaches to navigational planning use goal-directed heuristic search of problem spaces defined by spatial models of the navigation world [Fikes, Hart, and Nilsson 1972; Kuipers and Levitt 1988; McDermott and Davis 1984]. Since it employs a spatial model, we will refer to this family of methods as *model-based* planning. If the robot's navigation world is static and the robot planner has complete and correct knowledge of this world, then model-based planning guarantees both efficient processing and high-quality solutions [Aho *et. al* 1974]. However, complete world models are impossible to provide for operation in a dynamic environment, and current limitations of robot sensors and learning methods generally make it difficult to acquire such knowledge directly.

Experience-based reasoning [Hammond 1989, Kolodner and Simpson 1989, Riesbeck and Schank 1989, Sussman 1975, Winston 1982] offers an alternative approach to model-based navigational path planning. In this approach, the robot planner reuses previously formed plans to solve new planning problems: given a planning problem, the planner retrieves a past case of planning from its memory and adapts the plan stored in the case to meet the specifications of the current problem. This *case-based* family of methods promises several advantages over the model-based family [Kolodner 1993]: (i) since it relies on reusing specific experiences for solving new problems rather than reasoning from a general model of the navigation world, it provides for more efficient planning, and (ii) since it can start with relatively few cases in memory and dynamically acquire new cases based on the reasoner's interactions with the world, it makes few assumptions about the completeness and correctness of world knowledge. Note that, in theory, case-based planning offers these benefits without incurring any significant loss in the quality of solutions produced.

In the Router project, we have combined the case-based method with the model-based method for navigational planning. Since Router is a multistrategy system, it provides an experimental testbed for studying some trade-offs in multistrategy reasoning, and also for studying some of the basic hyptheses of case-based reasoning. In this paper, we report on a set of experiments that examine four hypotheses: (i) the case-based method is more efficient than the model-based method, (ii) the case-based method produces plans of quality equal to those produced by the model-based method, (iii) the case-based method requires less knowledge but has the same problem-solving coverage as the model-based method, and (iv) cases need to be decomposed into partial cases for efficient and effective problem solving. We find that while hypothesis (i) is true, the others are questionable. The goal of this paper is to report on these experiments, draw some lessons on the utility of the case-based and model-based methods for navigational planning, and the appropriateness of a specific framework for multistrategy reasoning.

General System Design

Router is a goal-directed multistrategy navigational path planning system. It operates in two kinds of navigation worlds: representations of office buildings such as the College of Computing Building (CoC) at Georgia Tech (GT), and representations of urban areas such as the Georgia Tech campus in Atlanta. In both worlds, the input to Router is a pair of spatial locations representing the initial and goal positions that the path should connect. The initial and goal locations are among the intersections between the pathways in the world; the pathways are the streets in the Georgia Tech domain and the corridors and hallways in the College of Computing domain. The output produced by the system is an ordered set of path segments (streets, hallways) between the initial and the goal locations. In addition, Router accepts as input feedback on the execution on a plan it produces and attempts to learn from both its successes and failures.

Router combines the model-based and case-based methods for planning navigation paths in these domains. It integrates these two methods in three dimensions: planning, memory, and learning. In the following subsections we briefly characterize the main components of Router in these three dimensions.

Planning Strategies

As in traditional robot planners, Router's model-based method performs a heuristic search of problem spaces defined by a spatial model of the navigation world. Unlike many traditional robot planners, however, Router's spatial model is only qualitative: it contains no quantitative information such as distances between locations. Further, the spatial model is organized in a neighborhood-subneighborhood hierarchy where a neighborhood pertains to a spatial region in the navigation world. The representation of a neighborhood contains information about the important pathways (streets, corridors) in the neighborhood, the directions of the pathways, the intersections between the pathways, the super- and subneighborhoods, and the relative locations of the subneighborhoods. A lower-level neighborhood in the hierarchy contains information about additional pathways and intersections but in a smaller region of space than a higher-level neighborhood. The model-based planner forms plans by a heuristic search with top-down control: it starts from the top-level neighborhood, uses direction as a heuristic, forms a high-level plan, and then sets up the subtask of plan refinement to fill in lower-level details.

Router's case-based method forms path plans by retrieving and adapting past planning cases. A case contains information about the initial and goal locations in a planning episode, the spatial neighborhoods to which the two locations belong, the path connecting the two locations, and whether the plan succeeded or failed upon execution in the world. It is indexed both by the initial and goal locations of the stored plan and by the neighborhoods to which the two locations belong. Given a planning problem, i.e., given the specification of the initial and goal locations in a navigation world, the case-based method uses the problem as a probe into its case memory. If the case memory contains a successful case whose initial and goal locations are the same as in the given problem, then the case directly provides the desired plan. If the case memory does not contain that a case that exactly matches the given problem, but contains a successful case such that the initial and goal locations of the case and the given problem are in the same neighborhoods, then the case is retrived and adapted. Case adaptation in Router consists of appending paths to one or both ends of a retrieved case.

Memory Organization

As indicated above, the hierarchically-organized spatial model of the navigation world in Router serves two purposes. First, it defines and decomposes problem spaces for search as described above. Second, it provides a scheme for organizing the case memory. The planning cases are organized around the neighborhood-subneighborhood hierarchy. Each case is indexed by the end locations of the path it contains, which serves as its primary index, and by the spatial neighborhoods of the end locations, which acts as its secondary index. In this way, the semantic and episodic knowledge in Router's memory are closely linked, with the semantic knowledge in the form of the spatial model providing the indexing scheme for organizing the memory of planning episodes or cases.

Multistrategy Planning

Given a planning task (or subtask), the model-based and the case-based methods offer alternative planning strategies. Router contains a simple introspective strategy selector that opportunistically selects a specific strategy for a given task. Note that the task may be the overall planning problem of a subtask set up by the model-based method (e.g., finding a top-level plan) or by the case-based method (e.g., adapting a case).

Learning

Router performs two kinds of learning. First, it acquires and stores new cases as it forms new plans and receives reports on their execution. A new case is automatically indexed both by the end locations of the path it contains and by the neighborhoods of the end locations as described above. Note that a case may contain a successful plan or a failed one depending on the feedback on the execution of the plan. Router keeps both successful and failed cases as in [Hammond 1989]. Note also that once a path is known, all of its constituent subpaths are also known, and so they too can be stored for their potential reuse in the future. In Router's domain, the decomposition of a case into sub-cases is obvious, and, if desired, the sub-cases can easily stored along with the cases.

Second, Router uses the new cases to learn the spatial model of its navigation world. For example, given a failed plan, it revises its world model to reflect the cause of the failure. In this paper, we focus only on speed-up learning in Router.

General Experiment Design

In order to evaluate the many dimensions, aspects and features of the theory behind Router, we conducted a series of ablation experiments. [Cohen and Howe 1988]. Although Router runs on a real robot called Stimpy, the experiments reported here were conducted on a simulated version of the robot so as to avoid the influence of new variables pertaining to perception and action. This allows us to focus on the theory behind Router.

Design of Experiment 1: The first set of experiments was designed to test two related hypotheses regarding the computational efficiency of case-based reasoning and multistrategy reasoning: H[1(i)] case-based reasoning is computationally more efficient than model-based reasoning, and, hence, H[1(ii)] integrated case-based and model-based reasoning is more efficient than model-based reasoning.

Design of Experiment 2: The second set of experiments was designed to test two related hypotheses regarding the quality of solutions produced by case-based reasoning and multistrategy reasoning: H[2(i)] case-based reasoning produces solutions of quality equal to those produced by model-based reasoning, and, hence, H[2(ii)] intergated case-based and model-based reasoning produces solutions of quality equal to those produced by model-based reasoning. Testing these hypotheses raises the issue of how to measure the quality of a solution. In the domain of navigational path planning, the logical answer is that a shorter navigation plan is better than a longer one. However, since Router contains no quantitative information (such as distances between locations), the issue becomes how to measure the shortness of a navigation plan. We used the number of path segments in a navigation plan for this purpose, where a path segment is defined as the path between two consecutive street changes in the overall path.

Design of Experiment 3: The third set of experiments was designed to test a hypothesis about the decomposition of cases into partial cases: H[3] the decomposition of cases into partial cases (at storage time) results in more efficient problem solving (on future problems). The *prima facie* justification for this hypothesis is that in general storing partial cases enables the retrieval of a case more appropriate to a given problem, and the retrieval of an appropriate case reduces the computational cost of adapting it to meet the specifications of the problem. Testing this hypothesis raises the issue of what is a reasonable partial case. In the domain of navigational path planning, the logical answer is that partial cases correspond to the path segments in a navigation plan (where, again, a path segment is defined as the path between two consecutive street changes).

Design of Experiment 4: The fourth set of experiments was designed to test two related hypotheses about the problem-solving coverage and knowledge requirements of case-based reasoning: H[4(i)] case-based reasoning can be bootstrapped with relatively few cases in memory, and H[4(ii)] case-based reasoning has the same problem-solving coverage as model-based reasoning even though its knowledge requirements are much smaller.

Router's two domains of GT campus and CoC building admit about 10000 and 1000 problems respectively. We conducted most of the above experiments with Router using 10 sets of 50 path planning problems each, where the problems and their order within a problem set were generated randomly. Since 50 problems may be too small a number for testing some of the above hypotheses, we conducted some experiments on a larger set of 1000 problems. All experiments were conducted on both the Georgia Tech domain and the College of Computing domain.

In all experiments involving multistrategy reasoning, the case memory was empty at the beginning of the experiment, but it grew as subsequent problem-solving cases were stored in it. In the experiments involving the exclusive use of case-based reasoning, the case memory had to be primed by adding (randomly formed) cases to the memory before conducting the experiments. Since we chose to generate the cases randomly, we did not make sure that there was a case going from each neighborhood to every other neighborhood. However, since we used a random method for generating the cases, in the runs with a large number of cases in memory at the start, the chances of there being a case going from a neighborhood to every other neighborhood were high. All experiments were conducted on a dedicated Sun workstation.

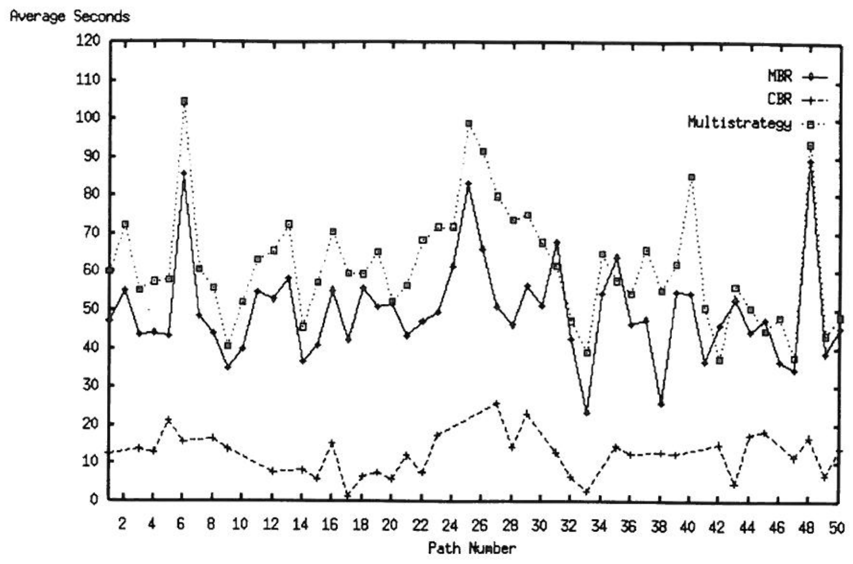

Figure 1: Comparison of average problem solving times for different strategies.

Experimental Data and Results

Results of Experiment 1: The data from experiment 1 confirms hypothesis H[1(i)]: it shows that when appropriate cases can be found in memory to perform (pure) case-based planning, case-based reasoning indeed is more efficient than either model-based planning or multistrategy planning (see Figure 1). In addition, this data shows that hypothesis H[1(ii)] is false: when the number of cases in memory is small (not shown in the figure), model-based planning performed faster than multistrategy reasoning most (approximately 92%) of the time while case-based planning typically fails to produce a solution at all.

When we repeated this experiment on a larger problem set of 1000 problems, we found that, for multistrategy reasoning, at the beginning the average problem-solving time increases as more problems are solved and more cases are stored in the case memory. In particular, the problem-solving time increased by approximately two hundredths of a second per problem for the first 1000 problems.

Results of Experiment 2: The data from experiment 2 shows that both hypotheses H[2(i)] and H[2(ii)] are false: in general, the case-based method produced solutions that were worse than the solutions produced by the model-based method, and, as a result, the intergated system also produced solutions of inferior quality. The model-based planner *always* produced paths with a smaller or equal number of path segments than the case-based planner.

Results of Experiment 3: The data from experiment 3 shows that hypothesis H[3] is false: the decomposition of cases into partial cases and the storage of partial cases increased the problem-solving time. On average, the problem-solving time for the entire path-planning process, including the storage of partial paths, was 1.7 times more than the problem-solving time without storage of partial paths (see Figure 2). (Because of this result, all other experiments were conducted without storing partial cases.)

Results of Experiment 4: The data from experiment 4 shows that while hypothesis H[4(i)] is true, hypothesis H[4(ii)] is false: while the case-based planner can indeed solve some problems with relatively few initial cases in the case memory, it covers fewer problems than can model-based reasoning. This data shows that the number of problems that can be solved by the case-based planner increases linearly with the initial number of cases in memory (see Figure 3). In particular, we needed to seed the case-based planner with approximately 16% of all the possible problems in our domain before it could solve approximately half (50of the problems given to it.

Theoretical Implications

The main theoretical implications of this work can be categorized along the lines of the four hypotheses we studied experimentally.

Problem Coverage and Knowledge Requirements: The result that we least expected from our experiments with Router is that the problem-solving

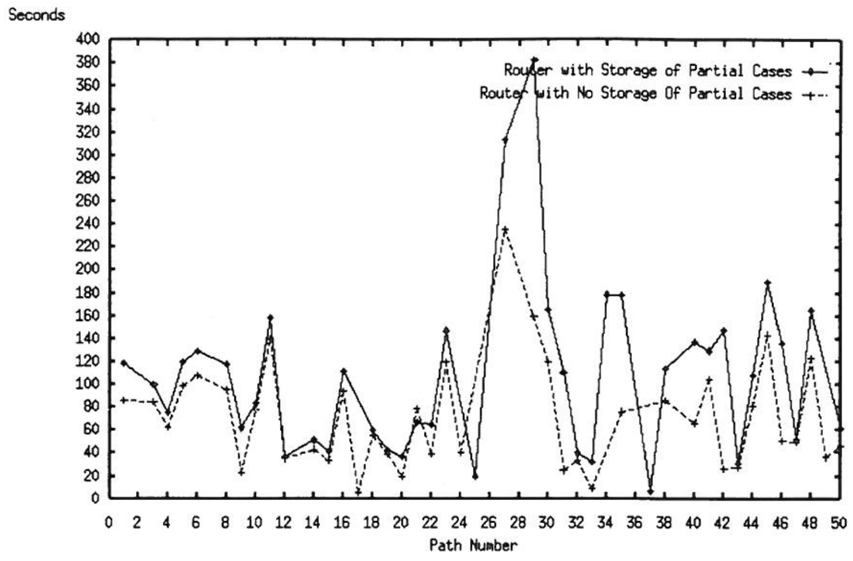

Figure 2: Effects of partial cases.

coverage of case-based reasoning increases only linearly with the number of cases in memory. We fully expected that the coverage of case-based planning will strongly depend on the number of cases in memory. However, initially we thought that as the case-based planner acquires new cases, the number of problems it can successfully solve will increase very rapidly — much more rapidly than the number of cases available in its memory. Clearly, this did not happen. Instead, we found that the case-based planner needed about 16% of all possible problems in its domain before it started working effectively in that it could solve a good fraction of the problems given to it, and even then it could solve only about half of all problems given to it. In general, it seems that for a domain with N elementary components and N(N-1)/2 possible problems, it is necessary to have 10*N cases in memory before the case-based planner can start working effectively.

As we argued in the introduction to this paper, model-based methods assume complete and correct knowledge of the world, and this assumption is not quite valid in many situations such as navigational path planning. However, it appears that the success of case-based methods is similarly dependent on the assumption that a large number of cases is available in memory. In fact, it seems that the problem-solving coverage of case-based reasoning is directly proportional to the number of available cases.

Computational Efficiency of Reasoning: The case-based method for planning appears to be computationally more efficient than the model-based method. This is because the case-based method reuses old plans rather than forming new ones.

However, integrated case-based and model-based planning can be computationally less efficient than either case-based or model-based planning. This appears to contradict the result reported by Veloso based on her work on the Prodigy system [Veloso 1992]. Prodigy integrates the methods of non-linear planning and derivational analogy. In her experiments with Prodigy in the domain of transport (logistic) planning, Veloso found that the combination of non-linear planning and derivational analogy was more efficient than the method of non-linear planning alone. We believe that this contradiction could be due to a number of factors: the differences between the derivational case-based reasoning used in Prodigy and transformational case-based reasoning used in Router, the differences between the non-linear planner in Prodigy and the model-based planner in Router, and the differences between the mechanisms for strategy selection in the two systems.

Quality of Solutions: In general, model-based reasoning appears to produce plans of a quality higher than does case-based reasoning. This contradicts the result reported by Koton based on her experiments with the Casey system [1988]. Casey performs medical diagnosis. It contains a global causal model of the heart as well as past cases of diagnosing some kinds of heart problems. It combines case-based reasoning and model-based reasoning for solving new heart-related problems. Koton reports that the combination of case-based and model-based reasoning produced solutions that were as accurate as those produced by the

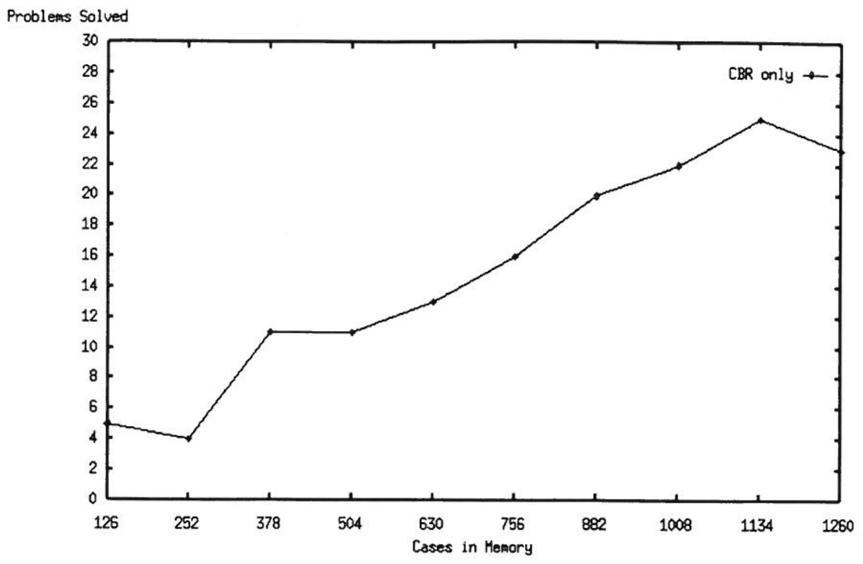

Figure 3: Number of problems solved (out of 50) with different amounts of memory bootstrapping for pure case-based path planning.

model-based method alone.

Given a diagnostic problem, Casey first uses its model of the heart to validate the apparently relevant cases in its memory, and then works with only the model-validated cases. Thus the quality of its solutions is due to its use of the model of the heart even when it is apparently using the case-based method for diagnosis. Therefore, the results from Casey should not be interpreted as implying that the use of the case-based method alone produces solutions of equal quality to those produced by the model-based method.

In a different version of our Router system, we have used Router's spatial model of its navigation world to (i) validate the cases retrieved from its memory, and (ii) validate the solutions produced by the case-based method by a kind of spatial simulation. The problem with this kind of model-based case validation is that it (again) assumes that the model is complete and correct. Kritik [Goel 1991] offers an alternative approach. It combines case-based and model-based reasoning for designing a class of engineering devices. Instead of using a global domain model, however, it uses case-specific device models. Its approach for integrating case-based and model-based reasoning offers the benefit of combining the efficiency of case-based reasoning with the accuracy of model-based reasoning without requiring global domain models.

Partial Cases: On the surface, the decomposition of cases into partial cases at storage time has the potential of retrieving more appropriate cases when a new problem is presented to the case-based planner at a later time. In turn, the retrieval of a case that more closely matches the new problem can help to reduce the computational cost of adapting the old plan to solve the problem. Our experiments with Router, however, clearly demonstrate that in general the use of partial cases significantly increases the cost of case-based reasoning. We fully expected that the use of partial cases will add to the cost of retrieving appropriate cases from memory because the memory would contain more cases. Our analysis of the Router experiments, however, indicates that the added cost of retrieval is small in relation to the added cost of decomposing the case into partial cases and storing the partial cases in memory.

An alternative design strategy might be to decompose a case into partial cases at case-adaptation time if so needed for problem solving. If a partial case is extracted during the process of modifying a retrieved case, then the partial case can be stored in the case memory without incurring the additional cost of case decomposition.

The above discussion leads us to the conclusion that at least for navigational path planning, an integration of case-based and model-based reasoning is the best strategy. An integrated approach would be able to combine the advantages of both methods in a complementary manner. In particular, such an integrated approach would combine the efficiency of cased-based planning and the robustness of model-based planning. However, it seems clear from our experiments that in order to fully exploit the advantages of the integrated approach for designing practical robot planners, Router's mechanism for strategy selection would need to be smarter.

References

A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley. 1974.

P. Cohen and A. Howe. How Evaluation Guides Research. *AI Magazine*, 9(4):35-43. Winter 1988.

R. Fikes, P. Hart, and N. Nilsson. Learning and Executing Generalized Robot Plans. *Artificial Intelligence*, 3:251-288. 1972.

A. Goel. A Model-Based Approach to Case Adaptation. *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, 143-148. Lawrence Erlbaum Associates. 1991.

K. Hammond. *Case-Based Planning: Viewing Planning as a Memory Task.* Academic Press. 1989.

J. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann Publishers. 1993.

J. Kolodner and R. Simpson. The MEDIATOR: Analysis of an Early Case-Based Problem Solver. *Cognitive Science*, 13, 507-549. 1989.

P. Koton. *Using Experience in Learning and Problem Solving.* PhD. Dissertation, Dept. of Computer Science, MIT, 1988.

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

D. McDermott and E. Davis. Planning Routes through Uncertain Territory. *Artificial Intelligence*, 22, 107-156. 1984.

C. Riesbeck and R. Schank. *Inside Case-Based Reasoning.* Lawrence Erlbaum Associates. 1989.

E. Stroulia and A. Goel. Functional Representation and Reasoning for Reflective Systems. *Applied Artificial Intelligence.* To appear. 1994.

G. Sussman. *A Computer Model of Skill Acquisition.* American Elsevier. 1975.

M. Veloso. *Learning by Analogical Reasoning in General Problem Solving.* Ph.D. Dissertation. Carnegie-Mellon University. 1992.

P. Winston. Learning New Principles from Precedents and Exercises. *Artificial Intelligence*, 19(3) 321-350. 1982.