From Models to Cases: Where Do Cases Come From and What Happens When A Case Is Not Available?*

Ashok K. Goel
College of Computing
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, Georgia 30332-0280
goel@cc.gatech.edu

Todd J. Callantine
Industrial and Systems Engineering
Georgia Institute of Technology
765 Ferst Drive
Atlanta, Georgia 30332-0205
tc@chmsr.gatech.edu

Michael W. Donnellan
Human Interface Technology Center
NCR Corporation
500 Tech Parkway, NW
Atlanta, Georgia 30313
mwd@ncrhitc.atlantaga.ncr.com

Andrés Gómez de Silva Garza
College of Computing
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, Georgia 30332-0280
andres@cc.gatech.edu

Juan Carlos Santamaria
College of Computing
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, Georgia 30332-0280
carlos@cc.gatech.edu

Abstract

The origin of cases is a central issue in cognitive models of case-based reasoning. Some recent work proposes the use of weak methods for generating solutions when a relevant case is not available, and chunking the solutions into cases for potential reuse. Our theory of case-based spatial planning and navigation suggests a different approach in which mental models of the world provide a way for solving new problems and acquiring cases. These mental models also provide a scheme for organizing the case mmeory, adapting old cases, and verifying new plans. The use of multiple methods, such as the case-based and model-based methods, raises another important issue in reasoning, namely, how to opportunistically select and dynamically integrate the methods. Our theory suggests the use of simple meta-reasoning to recursively select an appropriate method as the problem is decomposed into subproblems. This leads to the dynamic integration of different methods where one method is used for one subproblem and a different method for another subproblem.

*This work has been partially supported by a research grant from the Office of Naval Research (contract N00014-92-J-1234), a CER grant from NSF (grant CCR-86-19886), and equipment grants and donations from IBM, NCR, and Symbolics.

Goal and Motivations

Humans are often good at spatial planning and navigation. For example, most people navigate complex landscapes, cities, buildings etc., apparently with little difficulty. Further, their ability to navigate a given space appears to improve with experience, *i.e.*, they can form better navigation plans more efficiently. One computational theory for spatial planning and navigation comes from research on robotics. According to this theory, the robot's planner uses a mental model of the world to navigate through it, *e.g.*, (Fikes, Hart, & Nilsson, 1972), (Kuipers & Levitt, 1988), and (McDermott & Davis, 1984). This *model-based* method represents the navigation space in the form of a spatial model, and plans paths by a goal-directed heuristic search of the problem space defined by the model. While this combination of spatial models and goal-directed heuristic search has led to the development of some powerful robot planning and navigation systems, these techniques fail to explain how navigation ability

can improve with experience.

In the Router project, we have been developing an alternative *case-based* theory of spatial planning and navigation. An important issue in developing a case-based account of planning, or reasoning in general, is the *origin* of the cases: where do the cases come from? One answer to this question is that new cases are formed by adapting, perhaps combining, old cases. In fact, most case-based planning systems, such as Plexus (Alterman, 1988) and Chef (Hammond, 1989), give precisely this answer. An alternative answer is that cases result from solving problems by a different method. Both Prodigy (Veloso & Carbonell, 1991) and Priar (Kambhampati & Hendler, 1989), for example, use this strategy in generating cases. In both Prodigy and Priar, a nonlinear planner, based on means-ends analysis, generates solutions to a given problem and chunks the solution, along with its derivational trace, into a case. The cases are indexed by the goals they can help to achieve. Given a new problem, the agent searches its case memory to find whether a useful case is available, and if so adapts it to solve the problem, without appealing to the nonlinear planner. If a useful case is not available in memory, then the nonlinear planner takes over and solves the problem by itself.

Our work on the Router project suggests a different answer to the question of the origin of cases. Like Prodigy and Priar, it too hypothesizes that cases can often originate from solutions obtained by other methods in addition to arising from adaptations of old cases. However, it uses a model-based planner, like those used in robotics, for generating the solution. Router thus makes stronger assumptions about the available knowledge. Unlike Prodigy or Priar, the content of a case in Router includes the problem, the solution, and the outcome, but not the derivational trace. The case memory is organized around the world model. Like Prodigy and Priar, given a planning problem Router searches its case memory to find whether a useful case is available, and if so it adapts the case to solve the problem. Unlike Prodigy and Priar, however, Router uses simple meta-reasoning to integrate its model-based and case-based planners by opportunistically using the model-based planner for the task of case adaptation, and the case-based planner for similarly solving subgoals set up by the model-based planner.

Our theory, embodied in Router, explains how model-based planning can lead to the generation of cases, how it can be dynamically interleaved with case-based planning, and how an integrated planner's ability can improve with experience. This is one part of a larger theory in which case-based planning in turn can lead to the acquisition of mental models of the world. Our theory predicts that, whenever possible, people use case-based reasoning for spatial planning and navigation instead of reasoning from world models because the former method is generally more efficient than the latter. Our theory helps to explain some recent psychological data (Anderson, Kushmerick, & Lebiere, 1993), which indicates that people generally rely on their previous experiences in finding new path-plans. The theory also predicts that when a similar case is not available in memory, people reason from their world models, and chunk their solutions into cases for potential reuse. In addition, it predicts that people dynamically interleave model-based and case-based methods.

Task and Domain

Router plans paths on representations of two domains: the Georgia Tech campus and the College of Computing building. It takes as input an initial location and a destination location. These locations can be intersections between pathways, or the end-points of pathways. Pathways may be uni- or bi-directional, and more than two pathways may intersect at a given point. The output provided by Router is a path-plan from the initial location to the destination location, including directions of travel. The only constraint on the output is that the path-plan must be legal relative to the system's knowledge of the geographical space. If its knowledge is incomplete or incorrect, then the path-plan may well fail during execution. Router has access both to a spatial model of the navigation space and to previous path planning cases in the space. The spatial model represents qualitative knowledge about streets (or paths in general), their directions and their intersections. Streets are grouped into neighborhoods, and the neighborhoods are organized in a neighborhood-subneighborhood hierarchy. Higher-level neighborhoods contain knowledge of major streets while lower-level neighborhoods contain knowledge of minor and major streets within the neighborhood. Figure 1 illustrates a part of Router's spatial model of the Georgia Tech campus.

When Router solves a new path-planning problem, it chunks the specification of the problem, its solution, and the outcome (whether or not the plan succeeded upon execution in the world), and stores the case in memory. The cases are indexed by the specifications of the problems they solve, and are organized around the hierarchical spatial model. Once in memory, the newly acquired case is available to help the system solve future path-planning problems.

Reasoning Architecture

Router's architecture is comprised of five core processes:

Dynamic Memory: Router's memory is organized around the neighborhood-subneighborhood hierarchy of its spatial model. Each neighborhood in this hierarchy acts as an index to knowledge of the subneighborhoods and the streets in the neighborhood. It also acts as an index to path-planning cases whose initial and/or goal locations fall within the neighborhood. Router's

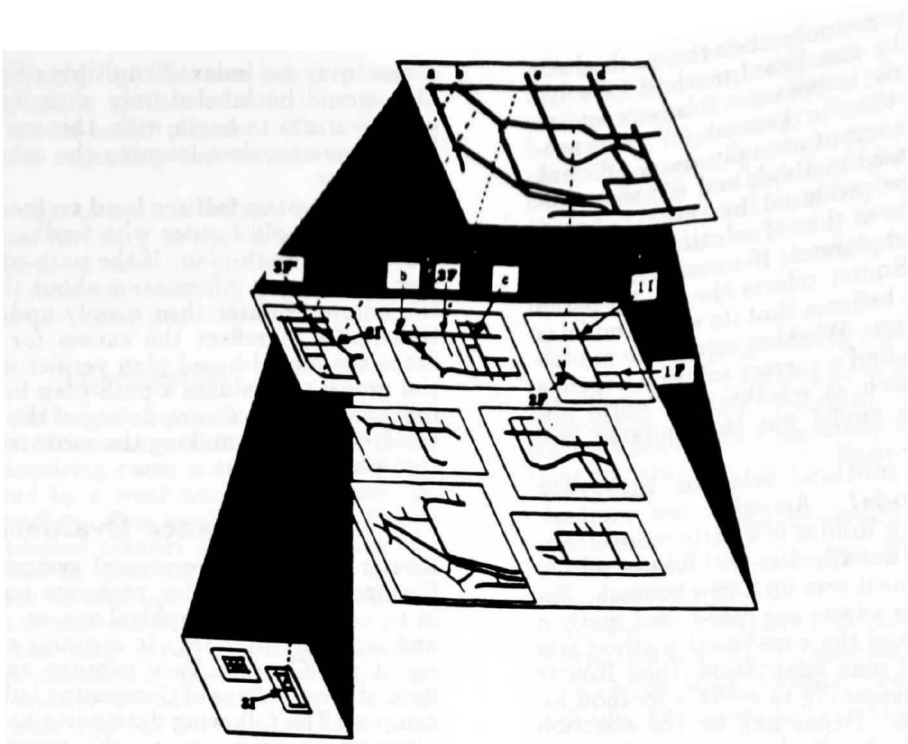

Figure 1: A Part of Router's Spatial Model of the GT Campus

memory is dynamic in that the system acquires new cases and learns indices to them.

Model-Based Planner: The model-based planner plans paths by searching the hierarchically-organized spatial model of the navigation space. The search is goal-directed, hierarchical and heuristic (Goel *et al.*, 1991). Each neighborhood in the spatial model defines a problem space, and the hierarchical organization of the model helps localize the search to specific problem spaces. The directions of pathways are used as a heuristic to help reduce the search space within neighborhoods.

Case-Based Planner: The "pure" case-based planner solves new problems by retrieving and potentially adapting previously planned paths. A path-plan, in the "pure" case-based mode, is adapted by *combining* with other path-plans retrieved from memory (Goel & Callantine, 1992). In the integrated planner, the model-based planner can also be used to adapt a case.

Model-Based Plan Verifier: The model-based plan verifier evaluates the correctness of a path-plan. The verifier uses the spatial model to simulate the proposed path-plan. If the path-plan fails, the method selector chooses either the model-based or the case-based method for repairing the failed path-plan by finding an alternative for the failed segment. Failure can occur, for instance, in situations in which the characteristics of the world have changed since a case was stored. Feedback from the user on the success of a path-plan is also verified before it is stored in memory.

Method Selector: Since Router has access to both cases and models, it must choose which reasoning method, case-based or model-based, to apply to each task at hand. It uses a simple meta-reasoner in the form of the method selector for this task as described below.

Opportunistic Method Selection

What are the criteria for selecting a method? Since Router can employ either the model-based or the case-based reasoning method to perform a path-planning task (or subtask), it needs some mechanism for choosing one method in any given situation. In general, the decision criterion may depend on several features, in addition to the nature of the task and the domain (Goel & Chandrasekaran, 1992): (1) properties that are desired of the solution to the task (*e.g.*, some measure of optimality); (2) properties that are desired of the method (*e.g.*, rapid execution); and (3) the availability of knowledge required by the various methods (*e.g.*, whether or not a useful case can be accessed from memory).

How does the method selector work? While all three factors may be important in general, the availability of knowledge (or lack thereof) is critical in guiding the decision process. Router uses a simple meta-reasoner for run-time method selection. It accesses a meta-memory of available methods each time a planning (sub)task has to be performed, and determines the usefulness of the applicable methods to the task. Then the meta-reasoner selects and invokes the most useful method according to the above criteria. For example, if a case similar to the current (sub)task is

available in the case memory, then the method selector may select the case-based method for solving the task. This is because Router's meta-reasoner believes that in general (i) the case-based method is computationally more efficient than the model-based method, and (ii) the quality of the solutions produced by the case-based method is the same as that of solutions produced by the model-based planner. If a case is not available in memory, Router selects the model-based method because it believes that its world model is complete and correct. Whether or not the model-based method can find a correct solution depends on many issues such as whether the world has changed since the model was last updated (see below).

How does the method selector integrate different methods? Actually, the method-selection process in Router is a little more elaborate than we have described so far. Router selects a method each time it sets up a new subtask. For example, if Router selects the case-based method at the top level, and the case-based method sets up the subtask of plan adaptation, then Router again uses meta-reasoning to select a method for adapting the plan. Depending on the selection criteria described above, this can result in the invocation of the case-based or the model-based planner. In this way, Router opportunistically integrates the two planning methods at its disposal.

Case Acquisition

Where do the initial cases come from? Router's case memory initially can be empty. If so, when the first few path-planning problems are presented to Router, the method selector has to choose the model-based method for solving them. The system chunks and stores the solutions generated by the model-based planner in the case memory. As it solves additional problems, its case memory grows and the method selector starts choosing the case-based planner for planning new paths. Router's reasoning thus gradually shifts from purely model-based to increasingly case-based. Alternatively, the user can directly supply an initial set of cases.

What cases are stored in memory? In addition to complete path-plans, Router also stores partial path-plans in its case memory. For example, if Router has found a path to go from intersection a to intersection z, say $a, b, c, d, ..., z$, then it stores the entire path as a case for potential reuse, since a future problem may require it to plan a path from a to z again. In addition, since the problem of going from any one intermediate location on the path to another (*e.g.*, from a to c, a to d, b to d and so on), may be required at some future time, the system also stores these "partial" path-plans as reusable cases.

How are the cases indexed? Cases are indexed both by their initial and goal locations and by the neighborhoods in the spatial model to which these locations belong. Since the neighborhoods in Router's spatial model can overlap, a case may be indexed multiply. Since a path-plan would be labeled only with its initial and goal locations to begin with, this multiple indexing scheme requires learning the other appropriate indices.

How does plan failure lead to learning? The user may supply Router with feedback on the execution of a path-plan. If the path-plan fails, the user may supply information about the causes for the failure. Router then simply updates its spatial model to reflect the causes for the failure. Since the model-based plan verifier uses the spatial model to evaluate a path-plan before reporting it to the user, this updating of the model helps the system avoid making the same mistake in future planning tasks.

Performance Evaluation

Router is a fully operational system written in Common Lisp. Router performs path-planning in two types of geographical spaces: urban areas and building interiors. It contains spatial models of the Georgia Tech campus and (the first floor of) the College of Computing building on the campus. The following data pertains to Router's planning performance on the representation of the Georgia Tech campus.

The streets in Router's model form 175 intersections and are organized in a 5-level neighborhood-subneighborhood hierarchy. This makes for a total of 15,282 distinct problems that the system can solve. We have tested Router on more than 50 sequences of 32 path-planning problems each. In each sequence, the system started with no cases in its memory but acquired them as it solved new problems. The system's performance can be characterized as follows:

(1) For sequences in which path-planning problems are presented in random order, Router's modality of reasoning shifts quite smoothly from model-based to case-based. That is, for the first few problems it uses the model-based method for path planning and chunks the solutions into cases. As the number of cases in its memory increases, it gradually starts using the case-based method for planning new paths.

(2) The types of problems (for instance, problems involving two subneighborhoods versus problems involving only a single neighborhood), the length of the paths planned, and the sequence in which the paths are planned and stored, all affect Router's subsequent performance. Problem sequences which begin with intra-neighborhood path-planning problems and then abruptly switch to inter-neighborhood problems do not provide a useful set of cases for solving later problems. For such problem sequences, Router does not display the smooth shift from model-based to case-based planning discussed above.

(3) On average, case retrieval and adaptation in Router is computationally less costly than model-based path-planning. Thus the system's average performance improves as the number of cases in its memory increases. Also, the quality of

solutions generated by the case-based planner appears comparable to that of solutions generated by the model-based planner. This verifies Router's beliefs regarding the usefulness of the two methods.

(4) Router is able to update its spatial model in response to user feedback on the failure of a path-plan. In addition, it uses the updated model for solving new problems, verifying proposed solutions, and thus avoiding past mistakes.

Discussion

The issue of the origin of cases is central to case-based models of human problem solving. One method for acquiring cases is to chunk the solutions generated by a weak nonlinear planner, as is done in Prodigy. Our work on Router shows that a model-based planner can also provide solutions that can be chunked into cases. It shows that the choice of alternative planning methods is not restricted to weak methods like those used by Prodigy but may involve stronger (knowledge rich) methods such as model-based planning. Further, the content of a case in Prodigy consists of a goal, a plan and the derivational trace. In contrast, the content of a case in Router consists of a goal, a plan for achieving it, and feedback on the outcome of the plan in the world.

Of course, we are not claiming that humans begin with complete or correct models and then acquire cases through model-based planning. In our general theory, models and cases coexist and complement one another. Given a partial model, model-based planning can help to acquire new cases. On the other hand, given a set of cases, the agent may generalize to form a mental model of the navigation space. In a sister project called Autognostic we are studying how models evolve from generalization over cases (Stroulia & Goel, 1992).

Our idea of combining case-based and model-based reasoning originates from earlier work on the Kritik system, which models design problem solving (Goel, 1991). Kritik's cases are designs of devices such as electric circuits and heat exchangers. Each case contains a model of the way the device's structural components produce its output behaviors. Given a new problem, Kritik looks for a similar case in memory and redesigns it by adapting it to the new problem's specification. The case-specific models help the system in diagnosing and adapting old designs to satisfy new specifications.

In cognitive models of problem solving, such as Prodigy, that combine case-based methods with other methods such as means-ends analysis, only one method is used to solve a given problem. That is, while these models use means-ends analysis to generate solutions and acquire cases, means-ends analysis plays no other functional role in case-based problem solving. In contrast, our work on Router shows how mental models can (and do) play several roles in addition to generating cases. For example, Router's spatial model also provides a scheme for indexing the case memory, for adapting retrieved cases to solve new problems, and for verifying planned paths. Perhaps more importantly, Router shows how simple meta-reasoning can be used to opportunistically select a method during reasoning so that one method is used to solve one part of a problem and another method is used to solve another part. This results in the dynamic interleaving of multiple methods.

Acknowledgements

We are grateful to Sambasiva Bhatta, B. Chandrasekaran, Mark Pearson, Murali Shankar, Nancy Vazquez, and especially Eleni Stroulia, for their contributions to the Router project.

References

Alterman, R. 1988. Adaptive Planning. *Cognitive Science* 12:393-422.

Anderson, J., Kushmerick, N., and Lebiere, C. 1993. Navigation and Conflict Resolution. In Anderson, J. *Rules of the Mind.* Hillsdale, New Jersey: Lawrence Erlbaum Associates, Inc. Forthcoming.

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

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

Goel, A., and Callantine, T. 1992. An Experience-Based Approach to Navigational Route Planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems,* 705-710. Raleigh, North Carolina.

Goel, A., Callantine, T., Shankar, M., and Chandrasekaran, B. 1991. Representation and Organization of Topographic Models of Physical Spaces for Route Planning. In *Proceedings of the Seventh IEEE Conference on Artificial Intelligence Applications,* 308-314. Miami, Florida.

Goel, A., and Chandrasekaran, B. 1992. Case-Based Design: A Task Analysis. In Tong, C., and Sriram, D. eds. *Artificial Intelligence Approaches to Engineering Design, Volume II: Innovative Design,* 165-184. San Diego, California: Academic Press.

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

Kambhampati, S., and Hendler, J. 1989. Flexible Reuse of Plans via Annotation and Verification. In *Proceedings of the Fifth IEEE Conference on Applications of Artificial Intelligence,* 37-44.

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

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

Stroulia, E., and Goel, A. 1992. An Architecture for Incremental Self-Adaptation. In *Proceedings of the Machine Learning 1992 Workshop on Computational Architectures for Supporting Machine Learning & Knowledge Acquisition.*

Veloso, M., and Carbonell, J. 1991. Automating Case Generation, Storage, and Retrieval in PRODIGY. In *Proceedings of the First International Workshop on Multistrategy Learning*, 363-377. Harpers Ferry, West Virginia.