An Integrated Experience-Based Approach to Navigational Path Planning for Autonomous Mobile Robots

Ashok Goel
College of Computing
Georgia Institute of Technology

Michael Donnellan
Human Interface Technology Center
NCR Atlanta

Nancy Vazquez
Distribution Resource Planning Group
AT&T Atlanta

Todd Callantine
School of Industrial and Systems Engineering
Georgia Institute of Technology

Abstract

Navigational path planning is a classical problem in autonomous mobile robotics. Most AI approaches to path planning use goal-directed heuristic search of problem spaces defined by spatial models of the navigation space. This paper explores an alternative approach that integrates a new case-based method with the traditional model-based method. Core issues in using case-based methods include the content, representation and indexing of past cases, organization of the case memory, retrieval of cases from memory relevant to the current problem, adaptation of retrieved cases to meet the specification of the current problem, and verification of the adapted solution to the problem. Our hypothesis is that spatial models of navigation spaces can provide answers to some of these issues in case-based path planning. The Router system examines this hypothesis in the context of path planning in geographical spaces. It uses a hierarchically-organized spatial model of the navigation space to index the cases and to organize the case memory. It also uses model-based reasoning to adapt past path-plans and to verify new ones. In addition, Router uses a flexible control architecture that allows for opportunistic selection and integration of the case-based and model-based planning methods. This paper provides an overview of the Router system.

Background and Motivations

The development of autonomous mobile robots apparently requires planning capabilities at several levels of spatial resolution. The Autonomous Robot Architecture (AuRA) [Arkin 1989], for example, identifies three levels of planning: *mission*, *navigation*, and *pilot*. The *mission planner* may take a set of goals as input (e.g., make a copy of this paper and collect my mail), and give a plan for achieving the goals as output (e.g., go to the copier room, make a copy of the paper, go the mail room, collect my mail, etc.). At the navigation level, a *path planner* may take a step in the mission plan as input. This input may be in the form of an initial location and goal location in the navigation space, (e.g., go from my office to the copier room). This path planner may give a path-plan from the initial to the goal location as its output (e.g., go from my office into the corridor, turn right, go to the end of the corridor, go through the door, turn right, etc). At the pilot level, a *motion planner* may take a step in the planned path as input (e.g., go from my office into the corridor), and may give a sequence of motor actions for accomplishing the goal as output (e.g., move left, move ahead, etc). A complete planning system for autonomous mobile robots would have the ability both to plan at different levels of spatial resolution and to handle interactions between planning tasks . In this paper, however, we focus only on path planning at the navigation level.

Since spatial knowledge, navigation, and path planning are among the classical issues in mobile

robotics, they have received considerable attention in artificial intelligence (AI) research related to robotics, for example [Fikes and Nilsson 1971] [Fikes, Hart and Nilsson 1972] [Kuipers 1978] [Kuipers and Levitt 1988] [Levitt and Lawton 1990] [McDermott and Davis 1984]. Most AI work on these issues 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 path planning and navigation systems, these techniques have some well-known limitations:

The Closed World Assumption

Most model-based approaches to path planning assume that the world is closed. That is, they assume, if only implicitly, that the robot's model of its microworld is correct, complete and consistent. The advantage of making this assumption is that it enables the planner to generate provably correct and/or optimal path-plans. However, the closed world assumption can be validated only for small, simple, and well-trodden microworlds; validation for large, complex or novel microworlds is infeasible. So the AI issue becomes: how might we relax the closed world assumption and design planners that can still produce "reasonable" path-plans?

Static World Models

Related to the closed world assumption, most model-based approaches to path planning use static models of the robot's microworld. That is, neither the content of knowledge nor the language of representations in these models changes over time. Such static models are useful only for microworlds that are themselves static over the lifetime of the planner. But the world in general is dynamic and constantly evolving. So the AI issue becomes: how might we design planners whose knowledge reflects changes in the robot's microworld? That is, how might we design path planners that can automatically *acquire* and *assimilate* new knowledge from the robot's interactions with its microworld?

Hand Coding of Knowledge

Related to the issue of static models, in most model-based approaches to path planning, the knowledge in the models is hand-coded by the system designer. Coding the models by hand is practical only when the robot's microworld is small, simple and static. If the microworld is large and complex, then coding the models by hand can be prohibitively expensive; if the world is dynamic, then hand coding the models is pragmatically infeasible. So, again, the AI issue is: how might we design planners that can automatically acquire and assimilate knowledge from the robot's interactions with its microworld?

Planning from First Principles

Most model-based approaches to path planning are based, if only implicitly, on the paradigm of planning from "first principles". These planners do not reuse the path-plans they create. Even if the planner is given exactly the same problem twice, on both occasions it solves the problem from the beginning. This repetitive calculation wastes precious computational resources. Such waste might be acceptable if the robot's microworld is small and simple, the computational resources available to the robot are large compared to its needs, and the temporal constraints on the robot's tasks are weak. However, as the microworld becomes large and complex, and the temporal constraints on the robot's task grow while the computational resources available to the robot do not, planning from first principles becomes infeasible. Thus the AI issue becomes: how might we design

path planners that can *reuse* previously planned paths?

Rigid Control Architectures

Related to the issue of planning from first principles, most model-based approaches to path planning use rigid control architectures. That is, the planner uses exactly the same reasoning method, inference mechanism, and control of processing regardless of the conditions of the microworld, the state of planning, or the availability of knowledge. In general, however, different microworlds present different opportunities and problems, and offer different types of knowledge. The AI issue becomes: how might we design planners that can *opportunistically* adjust their reasoning architectures to reflect changes in their state of planning and knowledge of the robot's microworld?

Over the last two years our research group has been investigating an *integrated experience-based approach* to navigational path planning for mobile robots. This integrated approach is based on the general idea of *model-based analogy* [Goel 1991a] in which domain models are used to adapt and combine past solutions to solve new problems. Our work is motivated by the issues listed above. It has so far led to four generations of an experimental system called Router, and several new prototypes are under development. A detailed description of the Router family of systems is beyond the scope of this paper. Instead, this paper provides a high-level overview of the latest operational version of Router called Router4.

The general goals and approach of the Router project can be characterized as follows:

1. Plan Reuse: One of the main goals of the Router project is to develop a mechanism for reusing previous solutions for solving new planning and navigation problems. Router's approach to plan reuse is based on the AI method of case-based reasoning [Kolodner and Simpson 1989]. In this method, new problems are solved by retrieving and adapting the solutions to similar problems encountered in the past. The new solution is stored in the case memory for potential reuse in the future. Case-based reasoning has previously been used for some types of mission planning, for example [Alterman 1988]. Router investigates the applicability of the case-based method for path planning at the navigation level in the AuRA architecture.

2. Grounding Case-Based Planning: The case-based method for planning raises a number of well-known issues [Hammond 1989] such as: how might the initial set of cases be acquired, how might the cases be indexed in memory, how might the case memory be organized, how might cases relevant to a new problem be retrieved from memory, how might retrieved cases be adapted to meet the specification of the problem, how might the proposed plan be verified, and how might a failed plan lead to learning of new knowledge? In our earlier work on case-based design of physical devices [Goel 1991a, 1991b], we discovered that causal models of the device domain provide the answers to many of these questions. In analogy to our earlier work on automated design, we hypothesize that the answers of many of issues in case-based path planning lie in the spatial models of robot's navigation space. Router4 investigates and substantiates this hypothesis.

3. Multi-Strategy Planning: One of our main hypotheses is that the case-based method complements but does not completely replace the traditional model-based methods for planning. The model-based approaches to path planning require a spatial model of the navigation space, while the case-based method requires a memory of past planning cases. Model-based methods generally

are more robust, while case-based methods can often be more efficient. Router4 investigates the integration of model-based and case-based methods for planning, exploiting the advantages and avoiding the drawbacks of the two methods.

4. Flexible Architecture: The use of multiple methods for planning raises the issue of method (or strategy) selection. Again, in our earlier work on automated design of physical devices, we discovered that task-method-subtask decomposition of the design task provides a flexible control architecture for selecting a method at design time [Goel and Chandrasekaran 1990]. In this decomposition, first the methods applicable to a task are identified, then the subtasks each method sets up are identified, next the methods applicable to each subtask are identified, and so on. The selection of a specific method to solve a given (sub)task depends on the properties of the applicable methods and the knowledge available in the domain. In analogy to our earlier work on automated design, we hypothesize that the task-method-subtask decomposition may provide a flexible control architecture for opportunistic method selection in planning. Router4 explores and substantiates this hypothesis.

5. Learning from Experience: Another main goal of the Router project is to develop a path planner that can learn from its experiences. More specifically, the goal is learn from the successes and failures of the planned paths. If the plan succeeds upon execution, the planner may store it in its case memory for potential reuse in future. If the plan fails, then the planner may reason upon the plan's performance, and adapt its knowledge and reasoning to reflect the causes for its failure. Current work on the Router project focuses on this issue of *performance-driven learning*.

Task and Domain

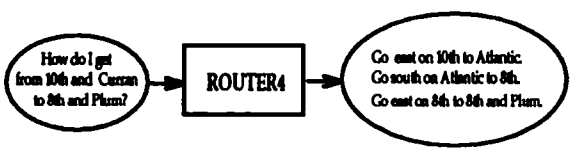

Figure 1: An Example of Router4's Task

Router4 plans paths in a representation of the Georgia Tech campus as described below. It takes as input a start location and a goal location as illustrated in Figure 1. The start and goal locations are intersections between pathways in geographical spaces of the kind shown in Figure 2. Pathways may be uni- or bi-directional, and intersections between pathways may allow left or right or both turns.

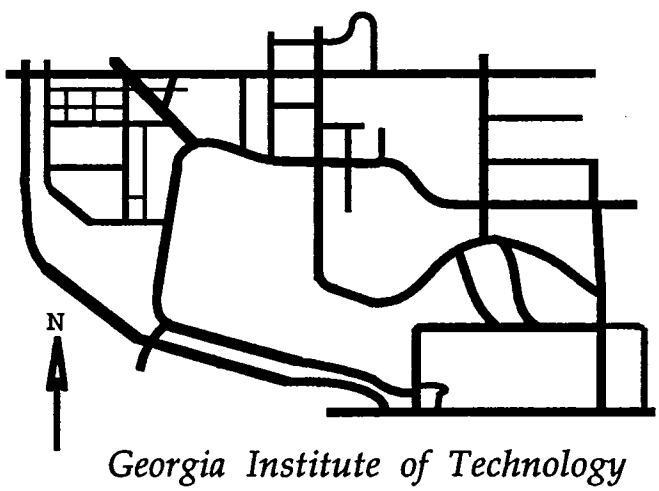

Figure 2: An Example of Router4's Domain

Router4's output is a path-plan from the start intersection to the goal intersection, including directions of travel as indicated in Figure 1. 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. Also, there is no requirement that the path-plan be optimal.

Neighborhood

Name: RICH
Sub-neighborhoods: None
Streets: (3rd 4th 5th Atlantic Brittan Cherry-2 Techwood)
Relative location to other neighborhoods:
(AECAL N)
(LIBRARY E)

Street

Name: TECHWOOD
Intersections: (5th 4th 3rd)
Directions: (N S)
Blocked-between: ()

Figure 3: Representation of Neighborhoods and Streets

Case

Type: Successful
Start intersection: (Curran 8th)
Goal intersection: (Hemphill Ferst-1)
Start neighborhood: Za
Goal neighborhood: Za
Path: (Curran 8th)(8th East until Hemphill)
(Hemphill South until Ferst-1)
(Hemphill Ferst-1)

Figure 4: Representation of a Case

Router4 has access to both prior domain knowledge and acquired experiences. Its domain knowledge is in the form of a spatial model of the geographical space. The spatial model represents qualitative knowledge about streets, and their directions and 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. The representation of neighborhoods and streets is shown in Figure 3.

When Router4 solves a new path-planning problem, it chunks the specifications of the problem and the solution into a case and stores the case in memory. The cases are indexed by the specifications of the problems they solve and are organized around the spatial model. Once in memory, the newly acquired case is available to help the system solve future path-planning problems. The representation of a case is shown in Figure 4.

System Architecture

Router4's architecture is comprised of one memory module, a natural language interface, and five core processes as indicated in Figure 5.

Memory: Router4'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.

Natural Language Interface: The natural language interface enables a user to interact with Router4 in quasi-English. The system accepts problem specifications and feedback about planned paths from the user, and reports on solutions to problems to the user. Feedback on path-plans comes in two flavors: successful path-plans, and failed path-plans including information about the causes for the plan failure. Successful path-plans result in updates to the case memory. In contrast, failed path-plans result in updates to the spatial model as well as to the case memory.

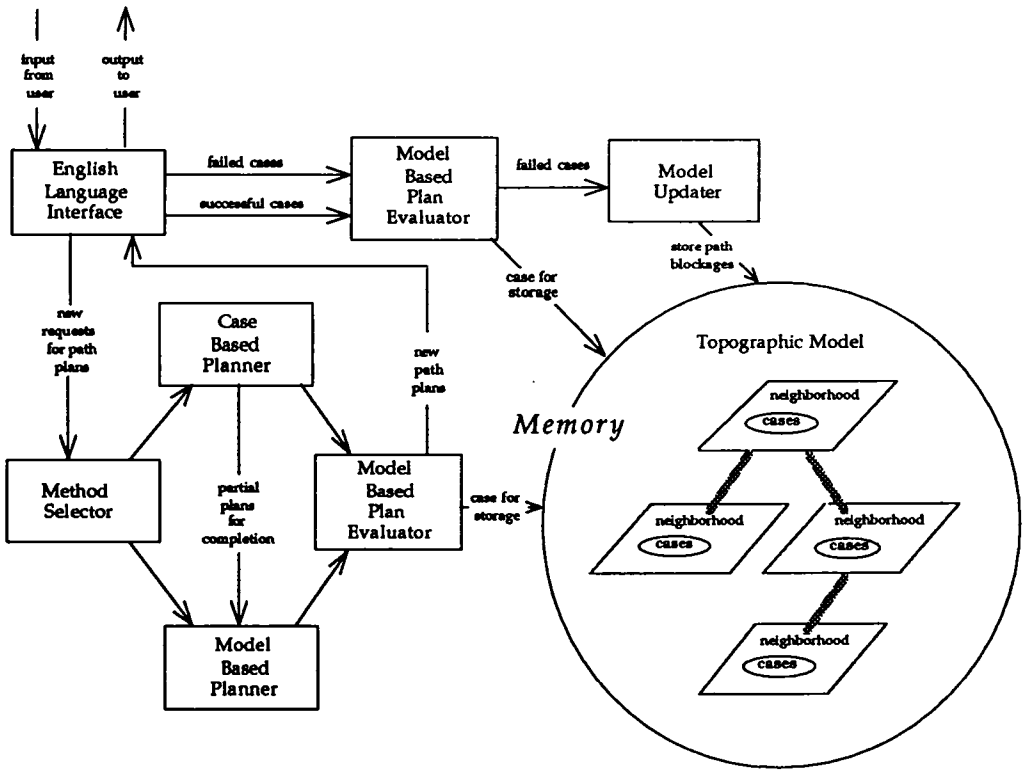

Figure 5: System Architecture

Model-Based Planner: The model-based planner plans paths by searching the hierarchically-organized spatial model of the navigation space as indicated in Figure 6. The search is goal-directed, hierarchical and heuristic [Goel, Callantine, Shankar and Chandrasekaran 1991]. The task of finding a path from the initial location to the goal location search is decomposed into a number of subtasks, such as finding a path from the neighborhood of the

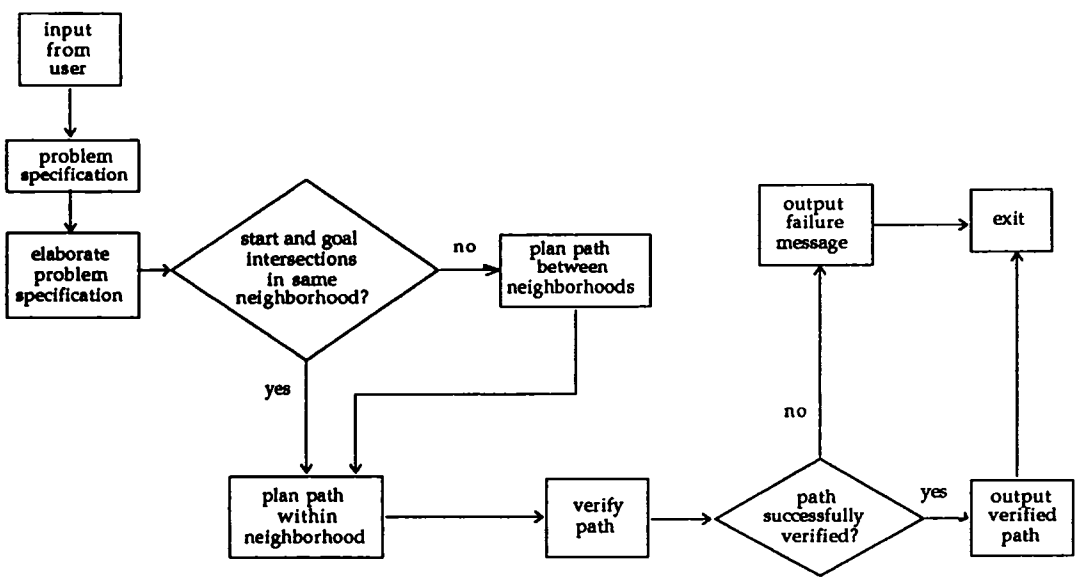

Figure 6: The Model-Based Planner

initial location to the neighborhood of the goal location, and finding connecting path segments within the neighborhoods of the initial and goal locations. Each neighborhood in the spatial model defines a compiled problem space, and the hierarchical organization of the model helps to localize the search to specific problem spaces. Direction of pathways is used as a heuristic to reduce the search space within neighborhoods.

Case-Based Planner: The "pure" case-based planner solves new problems by retrieving and adapting previously planned paths as indicated in Figure 7. A path-plan is adapted by retrieving and *combining* other path-plans [Goel and Callantine 1991a]. We discuss the case-based planner in more detail later in this paper.

Method Selector: Since Router4 has access to both cases and models, it must choose which reasoning method, case-based or model-based, to apply to the task at hand. It uses a simple meta-reasoner in the form of the method selector for this task. The method selector calculates an applicability measure for each method and chooses the one with the greater applicability value. The applicability measure takes into consideration the specification of the current task and the cases available in memory [Goel and Callantine 1991b]. This simple meta-reasoning is recursively applied to each subtask that Router4 sets up at run-time. As a result, Router4 may use one method for solving one part of a problem and the other method for solving another part.

Model-Based Plan Verifier: The model-based plan verifier evaluates the correctness of a path-plan before reporting it to the user. The verifier uses the spatial model to simulate the proposed path-plan. If the path-plan fails, then 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. The verifier validates user supplied reports on the success of a path-plan in the real world before storing it in the case memory; in addition, it validates cases retrieved from memory against the spatial model. The need for this type of validation arises because of possible changes to the spatial model based on feedback from the user.

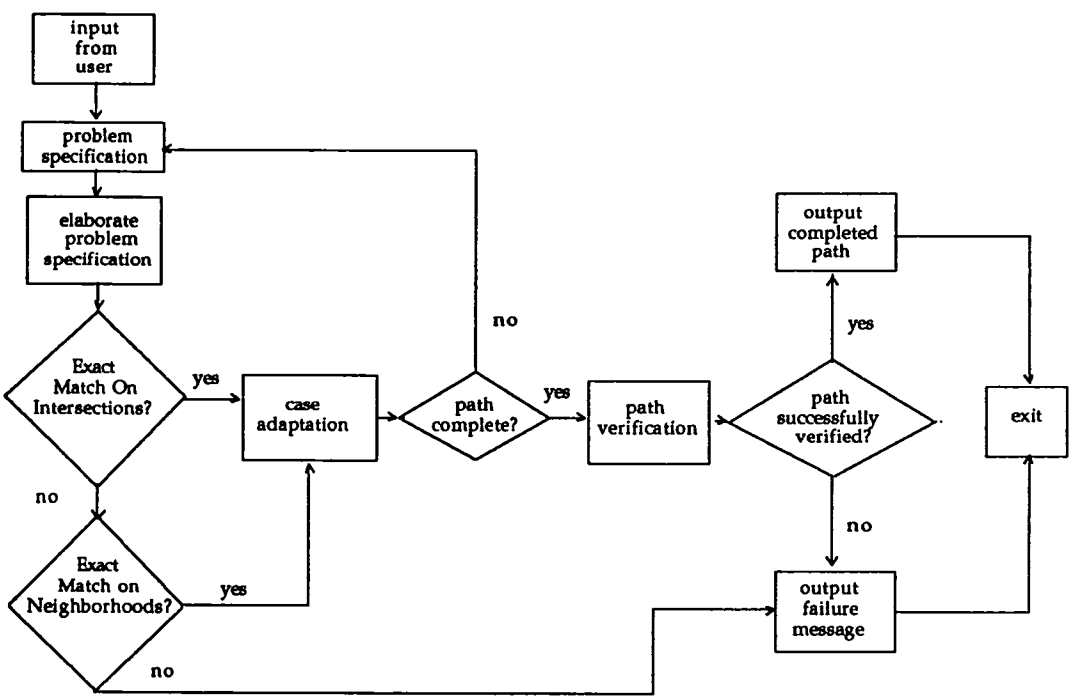

Figure 7: The "Pure" Case-Based Planner

Model Updater: The model updater is responsible for revising the spatial model when the user supplies feedback on the failure of path-plan in the real world. At present, Router4 has been evaluated for only one type of plan failure: a given pathway, which the spatial model showed as open between two intersections, actually is blocked. When the user supplies information about the failure of a path-plan due to the blockage of a pathway between specific intersections, the model updater simply notes this information in its representation of the street.

Cases and Models

We began this discussion by stating a hypothesis: that mental models can provide answers to some of the core questions in case-based planning. The following subsections describe Router4's answers to these questions.

Case Memory

How are cases indexed? Cases in Router4 are indexed by the neighborhoods in its spatial model. More specifically, neighborhoods act as indices to cases whose initial and/or goal locations fall within the neighborhood. In addition, cases are indexed by their start and goal locations. This secondary indexing scheme provides the case retriever with a finer-grain method for discriminating between potentially applicable cases.

How is the case memory organized? The neighborhood-subneighborhood hierarchy in the spatial model provides the organizational structure for the case memory. This model-based hierarchical organization of the case memory partitions and groups the stored cases into small sets of "similar" cases. This partitioning localizes case retrieval to the neighborhoods of start and goal locations in the specification of a new problem.

Path Planning

How are cases retrieved? Cases are retrieved based on their closeness to the start and goal locations in the specification of a new problem. In particular, the case retriever recalls similar cases by first identifying the neighborhoods which contain the problem's start and goal intersections and then selecting specific cases stored in these neighborhoods. The model-based plan verifier validates the selected cases against the current spatial model. In the current version of Router4, if a case is invalid then either an alternative case is retrieved or the control of processing is returned to the method selector and the model-based planner is invoked.

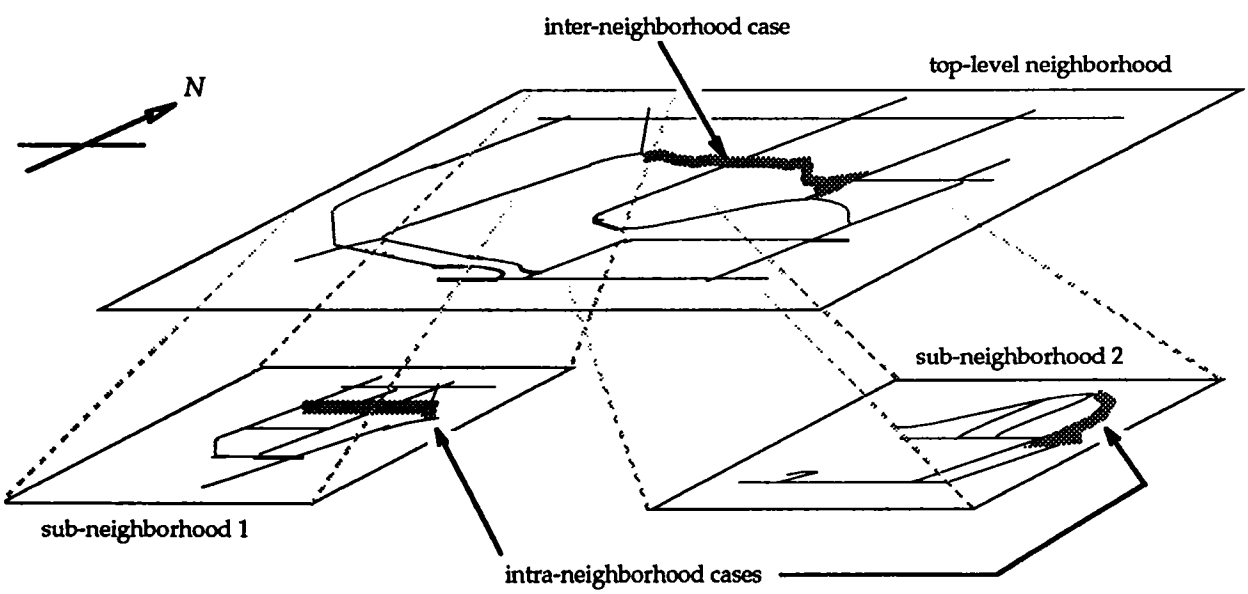

Figure 8: Combining Multiple Cases

How are cases adapted? The retrieved case is adapted by retrieving and combining additional cases. These additional cases solve the unsolved parts of the original problem as indicated in Figure 8. When a case that exactly matches the problem specification is not available in memory, the case retriever recalls the closest available case (e.g., the inter-neighborhood case in Figure 8), and the case adaptor recursively spawns new path-planning subtasks. The output of these new tasks supplies the missing parts of solutions to the original problem (e.g., the intra-neighborhood cases in Figure 8). In this way, Router4 combines the frameworks of subgoaling [Laird, Newell and Rosenbloom 1988] and case-based reasoning: the system spawns new subtasks whenever it fails to find an exactly matching case for solving a task, and attempts to solve the subtasks by case-based reasoning. This may be contrasted with the STRIPS system that similarly combined subgoaling and theorem proving [Fikes and Nilsson 1971].

The choice of a particular method for adapting a retrieved case depends on the specification of the adaptation task and the cases available in memory. The method selector in Router4 can also employ the model-based planner to supply the missing information needed for adapting a case. For example, the inter-neighborhood part of the solution in Figure 8 may be produced by the case-based planner while the intra-neighborhood portions are produced by the model-based planner. This leads to model-based analogy. In contrast to derivational analogy [Carbonell and Veloso 1988], model-based analogy uses domains instead of derivational records to adapt past cases to solve new problems.

How are plans verified? As described earlier, the model-based plan verifier evaluates the proposed path-plan by simulating execution of the plan on the spatial model.

Knowledge Acquisition

Where do the initial cases come from? Router4's case memory can initially be empty. If so, when the first few path-planning problems are presented to Router4, the method selector chooses the model-based planner for solving them. The solutions generated by the model-based planner are chunked and stored in the case memory. As the system solves additional problems, its case memory grows and the method selector starts choosing the case-based planner for planning new paths. Router4's reasoning thus gradually shifts from purely model-based to increasingly case-based. In addition, the user can directly supply the initial set of cases via the natural language interface.

How does plan failure lead to learning? The user may supply feedback concerning the execution of a path-plan through the natural language interface. If the path-plan fails, the user may supply information about the causes for the failure. Router4 then updates its spatial model to reflect the causes for the failure, as described earlier. 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 to avoid the same mistake in future planning tasks.

Performance and Evaluation

Router4 is a fully operational system written in Franz Allegro Common Lisp. The system uses Franz Allegro Flavors to provide support for its object-oriented design.

Router4's spatial model is an analogical representation of the real Georgia Tech campus. The streets in Router4's model form 96 intersections and are organized in a 3-level neighborhood-subneighborhood hierarchy. This makes for a total of 9,120 distinct problems that the system can solve. We have tested Router4 for 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, Router4'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 uses 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 Router4'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 sequences, Router4 does not display the smooth shift from model-based to case-based planning discussed above.

(3) On average, case retrieval and adaptation in Router4 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.

(4) Router4 is able to update its spatial model in response to user feedback on the failure of a path-plan. In addition, it reuses the updated model for solving new problems, verifying proposed solutions, and thus avoiding its previous mistakes.

(5) While Router4's ability to update its spatial model is useful, it carries a heavy computational cost. Checking for model updates during problem solving slows the system by at least a factor of 4 as compared to previous versions of Router which did not include the facility for model updating. This slowdown appears to be due to the specific implementation of Router4 rather than the conceptual framework underlying it.

Limitations

Router4's most glaring limitation is its relatively inflexible method of case adaptation. The system has no way of deciding whether all or just a part of a retrieved case is useful for solving a given problem, so it uses the entire path contained in a retrieved case in its solution. Router4 can thus produce suboptimal paths (that contain "zig-zags," for example).

Open Issues

Router4 is the fourth in a family of path-planning systems. Each generation in this family represents an attempt to address issues left unresolved from the preceding generation. Router1, following earlier AI work, represented the navigation space in the form of a hierarchically-organized spatial model and planned paths by heuristically searching the model in a top-down manner [Goel, Callantine, Shankar and Chandrasekaran 1991]. Router2 chunked the paths planned by Router1 into cases, stored them in memory, and reused them for solving new problems [Goel and Callantine 1992]. The case memory in Router2 was organized around Router1's spatial model. Router3 further integrated Router1 and Router2 into a flexible control architecture [Goel and Callantine 1991]. Router3 added a simple meta-reasoner in the form of a method selector that can opportunistically select between the model-based and case-based methods of path planning based on the current task and the available knowledge. It also used Router1's method for adapting the previous planned paths in Router2. Router4 integrates Router1 and Router2 even further. It also adds a simple capability for acquiring and assimilating knowledge. If and when the user of the system provides appropriate feedback on the failure of a planned path upon execution in the real world, Router4 attempts to revise its spatial model to reflect the causes for the failure. It also uses the spatial model to simulate and verify planned paths before delivering them to the system user. In addition, it provides a simple natural language interface for interacting with the user. Of course Router4 too leaves a number of open issues. Here we briefly discuss only those unresolved issues that we are presently exploring in a series of new projects:

1. Generality: Are Router4's knowledge representations, reasoning methods and control architecture general enough to be easily ported to other domains of path planning and navigation? To address this issue we are now evaluating Router4 in the domain of 3-dimensional spaces, such as the College of

Computing building at Georgia Tech. The preliminary results indicate that Router4's representations, methods and architecture are not limited to navigation spaces such as the Georgia Tech campus.

2. *Scalability:* Are Router4's knowledge representations, reasoning methods, and control architecture efficient enough to be easily scaled up to much larger domains of path planning and navigation? We do not presently know the answer to this question. However, in collaboration with our colleagues in the Database Systems group at Georgia Tech, we are planning to construct a much larger version of Router4 to test whether it is scalable.

3. *Distributed Processing:* Can Router4's knowledge representations, reasoning methods and control architecture admit parallelism and distribution in its processing? In collaboration with our colleagues in the Distributed Systems group at Georgia Tech, we are presently porting Router4 to a parallel and distributed environment. The preliminary results indicate Router4 not only admits parallelism and distribution at several levels of granularity but that its performance can improve considerably.

4. *Adaptability:* Are Router4's knowledge representations, reasoning methods, and control architecture easily modifiable to incorporate new methods? For example, can Router4 be easily modified to accommodate more effective methods of case adaptation? We believe that the answer to this question is affirmative. We deliberately designed the control architecture for Router4 to be flexible enough to incorporate and accommodate new methods as and when we identify them. To test this conjecture, we are presently adding new methods of case adaptation to Router4.

5. *Learning:* Do Router4's knowledge representations, reasoning methods, and control architecture admit more complex learning capabilities? Again, we believe that the answer to this question too is affirmative. Router4's learning capabilities currently are limited to automatically acquiring successful cases and automatically updating the spatial model for some very simple forms of plan failures. We are presently enhancing these capabilities. For example, our group is presently building another version of Router4 that generalizes route-planning cases. More ambitiously, we are presently developing a meta-Router system [Stroulia and Goel 1992] that contains knowledge of how Router4 plans paths. We expect that the meta-Router system, when complete, will be able to reason about the causes for the failures of the paths planned by the system, and to repair Router4's representations, methods and architecture. The preliminary results from these experiments are Router4's representations and architecture can easily admit several different types of learning.

6. *Real Robots:* Of course, the most important open issue from the Router4 experiment is whether the system can be used on a real autonomous mobile robot! Although embodying a Router-like planner in a real robot is precisely our long-term goal, we presently do not know the answer to this question. We are beginning to integrate Router4 into the Autonomous Robot Architecture (AuRA) [Arkin 1989] to which we alluded in the introduction. As a first step, we are extending Router4, which plans paths at the navigation level, for pilot-level planning in the AuRA framework.

Acknowledgements

The original Router team included B. Chandrasekaran and Murali Shankar. Andres Gomez and Juan Santamaria are building the Router5 system that extends Router4 to the domain of navigating office buildings and route planning at the pilot level. Sambasiva Bhatta, Mark Pearson and Eleni Stroulia are building the Distributed-Router system that exploits parallelism in Router4. Eleni Stroulia is also building the Meta-Router system that uses a meta-model of Router4 to learn from execution-time plan failures. This paper has also benefited from discussions with Ronald Arkin in the robotics group, Richard LeBlanc in the distributed systems group, and Sham Navathe in the database systems group at Georgia Tech. 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.

This research was conducted at Georgia Tech.

Contact: Ashok Goel
College of Computing
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, Georgia 30332-0280
Internet: goel@cc.gatech.edu
telephone: (404)-853-9371

References

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

[Arkin 1989] R. Arkin. Navigational Path Planning for a Vision-Based Mobile Robot. *Robotica*, 7:49-63, 1989.

[Carbonell and Veloso 1988] J. Carbonell and M. Veloso. Integrating Derivational Analogy into a General Problem Solving Architecture. In *Proceedings of the First Case-Based Reasoning Workshop*, Clearwater, FL, Los Altos: Morgan Kauffman.

[Fikes and Nilsson 1971] STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence*, 2(3/4):189-208, 1971.

[Fikes, Hart and Nilsson 1972] Learning and Executing Generalized Robot Plans. *Artificial Intelligence*, 3:251-288, 1971.

[Goel 1991a] A. Goel. A Model-Based Approach to Case Adaptation. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, Chicago, August 7-10 1991, pp. 143-148.

[Goel 1991b] A. Goel. Model Revision: A Theory of Incremental Model Learning. In *Proceedings of the Eighth International Conference on Machine Learning*, Chicago, June 27-29, 1991, pp. 605-609, Los Altos, CA: Morgan Kaufmann.

[Goel and Callantine 1991] A. Goel and T. Callantine. A Control Architecture for Run-Time Method Selection. *Proceedings of the AAAI-91 Workshop on Cooperation Among Heterogeneous Intelligent Systems*, Anaheim, California, July 1991.

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

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

[Goel and Chandrasekaran 1990] A. Goel and B. Chandrasekaran. A Task Structure for Case-Based Design. In *Proceedings of the 1990 IEEE International Conference on Systems, Man, and Cybernetics*, Los Angeles, California, November 1990, pp. 587-592.

[Hammond 1989] K. Hammond. *Case-Based Planning: Viewing Planning as a Memory Task*, 1989, Boston, MA: Academic.

[Kolodner and Simpson 1989] J. Kolodner and R. Simpson. The MEDIATOR: Analysis of an Early Case-Based Reasoner. *Cognitive Science*, 13:507-550, 1989.

[Kuipers 1978] B. Kuipers. Modeling Spatial Knowledge. *Cognitive Science*, 2(2):129-154, 1978.

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

[Laird, Newell and Rosenbloom 1987] J. Laird, A. Newell and and P. Rosenbloom. SOAR: An Architecture forGeneral Intelligence. *Artificial Intelligence*, 33:1-64, 1987.

[Levitt and Lawton 1990] T. Levitt and D. Lawton. Qualitative Navigation for Mobile Robots. *Artificial Intelligence*, 1990.

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

[Riesbeck and Schank 1989] C. Riesbeck and R. Schank. *Inside Case-Based Reasoning*, Hillsdale, NJ: Erlbaum,1989.

[Stroulia and Goel 1992] E. Stroulia and A. Goel. An Architecture for Incremental Self-Adaptation. In *Proceedings of the Machine Learning Workshop on Computational Architectures for Knowledge Acquisition and Machine Learning*, Aberdeen, United Kingdom, June 1992.