Route Planning by Analogy*

Karen Zita Haigh and Manuela Veloso
School of Computer Science
Carnegie Mellon University
Pittsburgh PA 15213-3891
khaigh@cs.cmu.edu, mmv@cs.cmu.edu

Abstract

There have been several efforts to create and use real maps in computer applications that automatically find good map routes. In general, online map representations do not include information that may be relevant for the purpose of generating good realistic routes, including for example traffic patterns, construction, or number of lanes. Furthermore, the notion of a good route is dependent on a variety of factors, such as the time of the day, and may also be user dependent. This motivation leads to our work on the accumulation and reuse of previously traversed routes as cases. In this paper, we demonstrate our route planning method which retrieves and reuses multiple past routing cases that collectively form a good basis for generating a new routing plan. We briefly present our similarity metric for retrieving a set of similar routes. The metric effectively takes into account the geometric and continuous-valued characteristics of a city map. We then present the replay mechanism and how the planner produces the route plan by analogizing from the retrieved similar past routes. We discuss in particular the strategy used to merge a set of cases and generate the new route. We use illustrative examples and show some empirical results from a detailed online map of the city of Pittsburgh containing over 18,000 intersections and 25,000 street segments.

1 Introduction

Case-based reasoning is a powerful problem solving technique which enables flexible reuse of experience. It is an on-going research challenge to apply case-based reasoning techniques to real-world domains. We have been investigating the use of case-based reasoning methods to enable a planner to reuse solutions to previous similar problems in order to solve new problems. In this paper, we demonstrate how our retrieval and adaptation techniques are applied and extended to the problem of route planning.

We are interested in having a computer generate good routes from an online representation of a map. Online representations of maps however do not usually include information that may be relevant for route planning, including traffic patterns, construction, one versus multi-lane roads, residential areas, time of the day, or a particular user's driving preferences. The path finding task is therefore dynamic and complex, and learning from route planning and execution experience is necessary.

Case-based reasoning methods allow us to take advantage of prior routing and execution experience. Our general approach for incorporating case-based reasoning with planning, execution, and learning within this real-world task consists of:

- Accumulating route planning episodes in a case library so that we can reuse previously visited routes and avoid unguided search.
- Retrieving a set of similar routes that collectively form a good basis for generating a new routing plan by using the geometric features of the domain.

*As in the proceedings of the International Conference on Case-Based Reasoning, Portugal, 1995; Springer Verlag

- Using execution experience to identify characteristics of particular routes that are not represented in the map and update the map and the *goodness* of the stored cases to reflect them.
- Using experience gained from altering plans during execution failures to acquire an understanding of when particular replanning techniques are applicable.

In previous papers, we presented several aspects of this project, including the similarity metric which effectively takes into account the natural geometric and continuous-valued characteristics of the map domain [4], and a discussion of the learning opportunities potentially offered by the real execution of the proposed planned routes [5].

The main focus of this paper is on presenting how the planning algorithm reuses the multiple similar retrieved cases. The paper is organized as follows. In Section 2 we show the representation of the real map, briefly introduce the planning domain itself, and discuss what information is represented and available to the planner. Section 3 reviews the storage and retrieval methods showing also how the goodness of a case is captured and used. Section 4 presents the analogical reuse of the multiple cases. Section 5 discusses empirical results and Section 6 describes related work. Finally, Section 7 draws conclusions on this work.

2 Representation of the Map Domain

Our current implementation uses the PRODIGY planner [1] and its analogical reasoning capabilities [11]. The route planning knowledge available to the planner consists of the map knowledge base, a set of operators (rules used to model changes in state), and the case library (described in the following section).

2.1 The Map

The map used in our work is a complete map of Pittsburgh containing over 18,000 intersections and 5,000 streets comprised of 25,000 segments. (An entire street is comprised of several segments corresponding to city blocks.)

The map is represented as a planar graph with the edges indicating street segments and the nodes indicating intersections. Associated with the intersections are the (x, y) coordinates of the intersection and a list of segments which meet at that intersection. Associated with each street segment is the name of the street containing it, and a range of numbers corresponding to building numbers on that block. In addition, there are several addresses of restaurants and shops in the city. Figure 1 shows a short excerpt from our files.

```
(intersection-coordinates i0 631912 499709)
(intersection-coordinates i1 632883 485117)
(segment-length s0 921 )
(segment-street-numbers s0 4600 4999)
(segment-intersection-match s0 i0 )
(segment-intersection-match s1 i0 )
(segment-street-mapping 0 S_Craig_St)
(address Great_Scot 413 S_Craig_St)
```

Figure 1: Excerpt from map database.

The representation, although it describes the map completely in terms of which streets exist, lacks in several areas important to an executing system. In particular, it does not indicate:

- existence of or direction of one-way streets,
- illegal turning directions,
- overpasses and other nonexistent intersections,
- traffic conditions,
- construction, and
- road quality, determined by factors such as number of lanes, surface (cobblestone, tarmac), and neighbourhood designation (residential, business).

This lack of information will lead to many situations in which the system needs to learn.

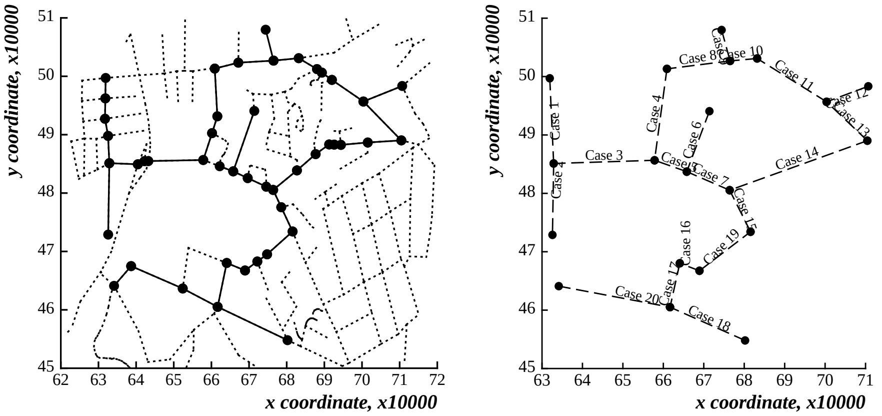
3 Storing and Retrieving Cases

A CBR planning system has to first identify cases that may be appropriate for reuse, and then modify them to solve the new problem. In order for the case identification phase to be efficient, the planning system must have a clear and easy method to store and subsequently retrieve past information. The following subsections describe our method for storing and retrieving routes. More detailed information regarding case retrieval (including run-time and efficiency results) can be found in a previous paper [4].

3.1 Case Representation and Indexing

When PRODIGY generates a plan, the detailed derivational trace of the successful planning episode is stored as a single case that can be multiply-indexed. Failed search decisions are annotated to be avoided at reuse time. For our route planning domain, the representation of each case will also include a detailed description of the situations encountered at execution time, including explanations of any errors that occurred and all replanning that was done to correct the problems.

Each case is also approximated by a set of straight line segments in a two-dimensional graph, and line segments are allowed to intersect only at their endpoints. The endpoints of these segments are generated at points where the new case intersects with existing cases as well as at points where the route changes direction and the segment would no longer approximate the route.



(a) Map. (b) Case graph representation of map.

Figure 2: Marked streets and intersections in the map (1% of the complete Pittsburgh map) are locations visited during previous planning.

This graph acts as an index into the case library so that cases can be easily retrieved. The resulting graph, which we call the *case graph*, is illustrated in Figure 2b. Figure 2a is a map in which solid line segments represent previously visited streets, and dotted segments represent unvisited streets. Figure 2b shows the abstract manner in which these paths are stored in the CBR indexing file. Note that Case 20 oversimplifies the path, but the bend in the road would not change the final routing (since there are no intersections along the route), so this abstraction is acceptable.

Note that several segments may together describe one complete case route, and one segment may index several cases.

3.2 Similarity Metric and Retrieval

Identifying cases relevant to the new problem is done by the use of a *similarity metric*, which estimates the similarity of cases to the problem at hand. An ideal metric might:

- take into account the relative desirability of different cases;
- suggest how multiple cases may be ordered in a single new solution; and
- identify which part(s) of a case are likely to be relevant.

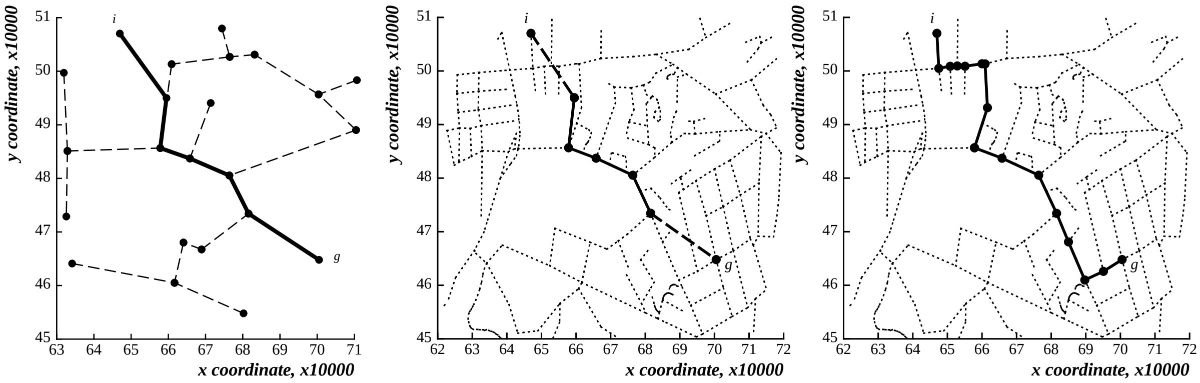
Finding a similarity metric that is both effective and fast is a difficult task for the researcher. It is sufficiently difficult that many existing CBR systems identify neither multiple cases nor partial cases at all. The metric developed by Haigh and Shewchuk [4] effectively takes into account the geometric and continuous-valued characteristics of a city map, and can generate multiple and partial cases.

Suppose we undertake to plan a route on our map from some initial location i to some goal location g. Although we want to reuse cases, we want to avoid long meandering routes and are therefore willing to traverse unexplored territory. It is important to find a reasonable compromise between staying on old routes and finding new ones. Hence, we assign each case an *efficiency* value β, which is a rough measure of how much a known case should be preferred to unexplored areasβ is an indicator of the "quality" of a case, and is independent of its length. After the case has been executed a few times, the β value associated with it will start reflecting the quality of the case as experienced in the real world. In particular, it will take into account traffic conditions, road quality and time of day. Each case-segment is annotated with information indicating what β should be as a function of the quality factors. For example, we might want to have a higher β value at rush hour:

if (15:00 $\leq$ current_time $\leq$ 18:00)
  then $\beta = 1.5$
else $\beta = 0.6$

Other possible comparisons might involve specific dates (e.g. construction), season or weather (e.g. impassability due to potholes, snow or mud), or direction (e.g. one-way streets). When invoked, the similarity metric uses the particular β associated with current conditions.

Since the only cases stored are those that the user executes, the metric will become biased towards routes that the user prefers.



(a) Path on case graph (b) Path on map (c) Final path

Figure 3: (a) The path found by the similarity metric in the case graph; dashed lines represent case edges, thick lines represent the path. (b) Path of cases superimposed on the real map; dashed lines represent where the planner needs to plan from scratch, solid lines represent cases. (c) Path modified by PRODIGY/ANALOGY to conform to real world constraints.

Figure 3 shows a path chosen by the similarity metric between the labelled initial (i) and goal (g) points and some β value assigned to each case in the case graph. Each segment in the path of Figure 3a corresponds to a case (or more than one) that the similarity metric believes will be helpful in solving the new problem. This path can also provide the planner with useful hints about how to link the cases together: where to change from case to case, as well as when to leave the cases entirely and start planning from scratch.

Note that a routing case is not used in reverse, since in general good routing situations are directional (e.g. one-way streets or traffic conditions). Note also that the path given to PRODIGY/ANALOGY would not be executable in the real world because it traverses several regions where there are no streets. It is the planner's job to knit this information together into a plan, taking into account details such as one-way streets and illegal turns that cannot be resolved by the geometric algorithm. This process is described further in the following section.

- Input: The map description; the initial location; the goal location; and an ordered list of (chopped) cases $C_1, C_2, \ldots, C_n$ (each C_i consists of a sequence of relevant steps to the new situation).
- Output: P, a route from the initial to the goal locations.

procedure *sequential-analogical-replay*
1. $i \leftarrow 1$; $j \leftarrow 1$ (*i is the guiding case; j is the guiding step in the case*)
2. All case steps are marked usable.
3. Terminate if the goal location was reached.
4. Get the j^{th} plan step from the case C_i, i.e. C_i^j.
5. Validate the case choice C_i^j.
6. If the case choice C_i^j is invalidated,
7.   then: Mark unusable the case steps strictly dependent of C_i^j.
8. $j \leftarrow$ next usable step in C_i
9.   Plan by searching on the map (heuristic or iterative deepening).
10.   If the new plan step matches some usable case step C_k^l,
11.   then: Mark unusable the steps in cases C_m, $i \leq m < k$, if $k > i$.
12.   Mark unusable the case steps C_k^m, $1 \leq m < l$, if $l > 1$.
13. $i \leftarrow k$; $j \leftarrow l$.
14.   Go to 4.
15.   else: Add the new step to plan P.
16.   Go to 3.
17.   else: Add the new step C_i^j to plan P.
18.   Link the new plan step to the case step.
19.   Advance case C_i to its next step: $j \leftarrow j + 1$.
20.   If the end of case C_i was reached,
21.   then: $i \leftarrow i + 1$; $j \leftarrow 1$.
22.   Go to step 2.
23. Return the plan P.

Table 1: Sketch of Serial Analogical Replay of a Sequence of Cases.

4 Route Planning by Analogy

We follow the analogical replay strategy developed by Veloso [11] in PRODIGY/ ANALOGY. The replay technique involves a closely coupled interaction between planning using the domain theory (operators and other static knowledge of the world) and modification of a set of similar cases.

The cases are derivational traces of both successful and failed decisions in past planning episodes, as well as the justifications for each decision. The case replay mechanism of PRODIGY/ANALOGY involves a reinterpretation of the justifications for case decisions within the context of the new problem, reusing past decisions when the justifications hold true, avoiding failed decisions, and replanning using the domain theory when the transfer fails. The general PRODIGY/ANALOGY replay algorithm is domain-independent and is designed to replay multiple cases, and is capable of merging cases in several different manners. (Merge modes include sequential, interleaved, guided and random. [10].) The replay procedure provides guidance to the general choice points of the planner.

In this domain, the retrieval procedure returns a list of cases, $C_1, \ldots, C_n$, ordered according to the sequence in which the metric believes they should be reused. The geometric processing of the similarity metric also identifies which parts of each case should be used in the generation of the new solution, and therefore only the relevant steps are handed to the replay procedure.

The set of cases returned by the retrieval procedure should be merged by the replay mechanism to form the solution route to a single one-goal problem[1]. In this sense, the use of multiple cases in this domain

[1] A single problem can consist of multiple goals, but the similarity metric is only called for a single source/goal pair.

differs from the use of multiple cases in other planning domains where different cases cover different top-level goals [11].

In the route planning domain, PRODIGY/ANALOGY is set to use a sequential merging strategy to combine the cases. Because of the partial match between the cases and the new situation, the replay algorithm is prepared to do any needed extra planning. In particular, any extra planning to *connect* a pair of cases is done by an iterative deepening search based on a estimated depth bound. Informally, at the end of each case, the replay algorithm proceeds by searching carefully for the next case. The illustrative example in the empirical section shows the effectiveness of this merging strategy.

An equivalent search process is performed if a case step becomes *invalidated*, for example, in a situation where the past case uses a segment (e.g. a bridge) that is not available in the current map. The map is updated by executions to reflect the change (see [5]) while the past cases still refer to the unavailable segment. We do not alter the cases since it could be computationally expensive in a large case library; instead we modify the β values of the case to reflect its poor quality. In addition, the case might become relevant at a later date.

Table 1 sketches the replay algorithm with this sequential merging procedure. We present the serial processing of the sequentially-ordered multiple cases.

The adaptation in the replay procedure involves a validation of the steps proposed by the cases. When there is a need to diverge from the proposed case steps (step 9), note that the algorithm tries to return to the cases "as soon as possible" (see step 10, in which the algorithm tries to match a newly generated step with a case step, even when not the immediately obvious next one). This bias in the sequential merge combined with the β-biased similarity metric allows an interesting reuse of good quality previously visited routes.

5 Experiments

The experiments we describe in this section illustrate the use of our similarity metric combined with the sequential analogical replay of multiple cases. The ultimate goal of our work is to use real planning and execution cases to learn from and produce reliable routes. The experiments here are focused on the topic of this paper and do not address the complete goal of our work.

It has been claimed that complex adaptation strategies can be dangerous because it is difficult to guarantee a good final solution [6]. Our goal in these experiments, therefore, was to show that our method of selecting similar cases and subsequently merging them produces reasonable routes to new problems. The experiments we describe have been performed on a portion of the real map of Pittsburgh with approximately 18,000 intersections.

We randomly generated a set of 30 problems, and solved them using (a) A* to find the physically shortest path, (b) a heuristic based on a local desire to head in the direction of the goal (which corresponds to what someone might do while driving in unfamiliar territory), and (c) PRODIGY/ANALOGY using as a case library the combined set of A* and heuristic from the other 29 problems. In practice, we would prefer to accumulate cases from everyday planning and execution experience; something hard to capture in a heuristic function.

	Time (s)	Nodes	Route Length
A*	4881	1067	119567
Heuristic	686	167	153401
Analogy	604	155	129132

Table 2: Average time, number of nodes expanded and (Euclidean) length of solution for the three search algorithms.

Looking at the results from Table 2, it is clear that finding an optimal route is extremely expensive when compared to either the local direction-based heuristic or to analogy. Given that the improvement in solution length is only 22% and 8% respectively, it is hard to justify spending more than 600% of time and resources to find the optimal solution. Note that Analogy has less search to do than either of the other algorithms because the majority of the search was done in previous problem-solving episodes.

PRODIGY/ANALOGY is quite capable of handling multiple goals in this domain.

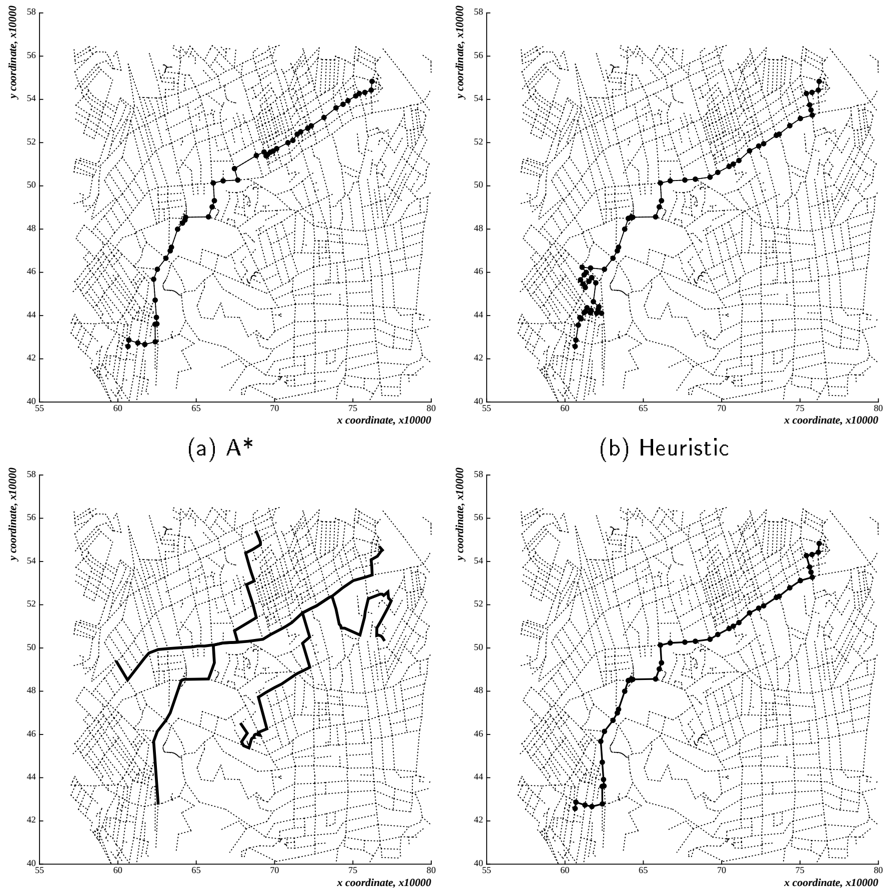
Figure 4 shows an example of one problem as it was solved by A*, PRODIGY/ANALOGY and the heuristic.

The A* route is the shortest route between the initial and goal points, however, it goes through a tightly congested residential area, rather than taking the larger street to the south[2]. In terms of convenience, the optimal route is in fact *worse* than either the heuristic- or analogy-generated routes. This situation occurs very often in real world cities because drivers select routes based on their convenience, familiarity and reliability rather than optimal length.

The heuristically generated path does very little search as it does no backtracking. Although reasonable in this situation, it can get sidetracked in non-grid-like cities, and even in this example meanders towards the goal. Several of the routes were distinctly unreasonable, more than doubling the length of the optimal route.

The similarity metric selected three relevant cases in the case library (Figure 4c). The metric examined all available routes (those generated by both A* and the heuristic) and selected those which described the new problem most effectively. Note that case 0 eliminates the meandering of the heuristic path, while case 5 selects the better route near the initial point. The metric then returned the relevant parts of each case to the reuse algorithm, which faithfully followed the cases, adding the additional steps at the beginning and

[2] For those readers familiar with Pittsburgh, the residential area is Shadyside, and Fifth Avenue is the larger street.



(a) A*

(b) Heuristic

(c) Selected Cases

(d) Analogy

Problem: initial location (762213,548347) to goal location (606469,425757)

Case 0 (A*): initial location (688451,553663) to goal location (625678,428191)

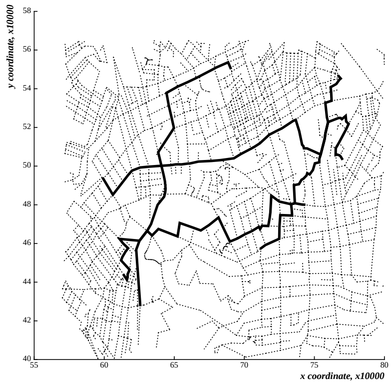
Case 4 (H): initial location (599257,493531) to goal location (769737,503789)

Case 5 (A*): initial location (678681,464750) to goal location (767392,546448)

Figure 4: (a) The path as generated by A*. (b) The path as generated by a search heuristic. (c) The cases selected as being relevant to the current problem. (d) The path as generated by analogy.

end of the cases and successfully switching between the cases along the route.

Based on these observations, we believe that an algorithm more reactive than A* is needed. Storing routes actually executed by the driver is a good way to bias the system towards routes that he prefers. The loss of physical optimality will be regained as the system increases its knowledge of the convenience and reliability of the route through experience. Solutions generated by analogy will therefore be better than generating solutions from scratch. We are currently implementing a model of the execution domain to test this theory[3]. To date, the β values output by the model have been accurate enough for the similarity metric to correctly identify two different routes for the problem described in Figure 4. Figure 5 shows the four relevant cases. During rush hour, when larger streets are more congested, the similarity metric selects the lower route through a park and residential areas. At other times, it selects the route along larger roads.



	Initial	Goal
Problem:	(756158,537424)	(734187,454947)
Case 0:	(614040,443632)	(742726,480067)
Case 4:	(599257,493531)	(769737,503789)
Case 15:	(711896,457519)	(767392,546448)
Case 26:	(625678,428191)	(690159,550707)

Figure 5: The four relevant cases under different conditions for the problem described in figure 4.

Through our on-going experiments, which we briefly illustrated above, we have demonstrated that our method of retrieving and reusing multiple cases produces solutions that are reasonable and desirable, *i.e.* they correspond to familiar routes (the cases) and are not overly sinuous or long. With the learning methods described previously (see [5]), the map information and the knowledge about the quality of each case improves with experience.

6 Related Work

Most robotics path planners (e.g. Dyna [7], COLUMBUS [9], Xavier [3], NavLab [8]) don't remember paths or their quality, and typically use shortest path, dynamic programming or decision theoretic algorithms to determine routes. We do not believe these algorithms are sufficient in this domain for several reasons, including:

- they typically examine the entire map, doing a blind search ignoring the general direction of the goal, and in a large map will be very slow;
- they are unable to interleave planning with execution or deal with unexpected situations;
- they can not distinguish between multiple routes of equal length but different quality; and
- they are strictly shortest-path algorithms, not considering path reliability or convenience factors such as time-to-goal, road quality, or user preferences.

ROUTER (developed by Goel *et. al* [2]) and R-Finder (developed by Liu *et. al* [6]) are the only other case-based route-planning systems the authors are aware of. However they both have extremely simple retrieval and modification algorithms, considerably reducing the transfer rate of prior experience. In addition, they do not re-validate the plans in the up-to-date map, making it more likely that in a dynamic world they will return invalid solutions; nor do they remember the quality of cases in an attempt to improve retrieval and the quality of the plans they generate. Finally, the goal of their work is to speed up planning; we feel that

[3] Originally, we had planned to execute this on a real robot, but collecting sufficient data would have been too expensive.

1. Given a new problem, find a set of similar cases, using stored β values.
2. Modify case(s) into new plan.
3. Execute plan on real robot. (*Not implemented yet.*)
4. If execution of plan is successful:
   Then: Add new case to library.
   Assign appropriate β values.
   Otherwise: Identify reason for failure. (*Not implemented yet.*)
   Modify world knowledge as applicable.
   (β values, map)
   Add any successful parts of plan to case library

Figure 6: Our integrated planning and learning route-planning algorithm. Current status of the various stages are marked in italics.

fast planning can be achieved by a combination of Dijkstra's algorithm and goal-oriented heuristics, and therefore our focus is instead on the more difficult questions of plan quality and reliability.

7 Conclusion and Discussion

In this paper, we have described our approach to applying case-based reasoning methods to route planning. We motivated the need for an integrated system that incrementally acquires experience. We briefly presented a similarity metric that takes advantage of the geometric characteristics of the map and returns a set of similar previously traversed routes that are jointly relevant to the new situation. Our metric supports the need for more naturalized mechanisms for detecting similar cases. We discussed how the system creates a new route plan by merging guidance from the multiple retrieved cases and planning from the map description and the routing operators. We presented the analogical replay algorithm using a sequential technique to merge the multiple similar retrieved cases.

We believe that it is important to create good quality, reliable plans, and that using familiar well-indexed routes, shown to be successful in the past, is an effective way to do this. As our empirical results have shown, optimal algorithms seem far too costly for the benefits gained, while heuristically-based algorithms may get led astray.

The entire integrated planning and learning algorithm is summarized in Figure 6. Except where marked otherwise, it is fully implemented and running. The code is available on request. We are currently extensively evaluating our work and its impact in realistic route planning situations using the complete online map of the city of Pittsburgh.

We believe that any system that interacts with the real-world will have to deal with a changing environment. We hope that our system, with its robust integration of planning, case-based reasoning, and learning from real execution, will form a good basis for future exploration in this area.

References

[1] Jaime G. Car bon ell, and the PRODIGY Research Group. PRODIGY4.0: The manual and tutorial. Technical Report CMU-CS-92-150, School of Computer Science, Carnegie Mellon University, June 1992.

[2] Ashok Goel and *et al.* Multistrategy adaptive path planning. *IEEE Expert*, 6(6):57–65, December 1994.

[3] Richard Goodwin and Reid Simmons. Rational handling of multiple goals for mobile robots. In J. Hendler, editor, *Artificial Intelligence Planning Systems: Proceedings of the First International Conference (AIPS92)*, June 1992.

[4] Karen Zita Haigh and Jon athan Richard Shewchuk. Geometric similarity metrics for case-based reasoning. In *Case-Based Reasoning: Working Notes from the AAAI-94 Workshop*, pages 182–187, Seattle, WA, August 1994. AAAI Press.

[5] Karen Zita Haigh, Jon athan Richard Shewchuk, and Manuela M. Veloso. Route planning and learning from execution. In *Working notes from the AAAI Fall Symposium "Planning and Learning: On to Real Applications"*, pages 58–64, New Orleans, LA, November 1994. AAAI Press.

[6] Bing Liu and *et al.* Integrating case-based reasoning, knowledge-based approach and Dijkstra algorithm for route finding. In *Proceedings of the Tenth Conference on Artificial Intelligence for Applications*, pages 149–55, San Antonia, TX, March 1994.

[7] Richard S. Sutton. Planning by incremental dynamic programming. In *Machine Learning: Proceedings of the 8^{th} International Workshop*, pages 353–357. Morgan Kaufmann, 1991.

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

[9] Sebastian B. Thrun. Exploration and model building in mobile robot domains. In *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, March 1993.

[10] Manuela M. Veloso. Automatic storage, retrieval, and replay of multiple cases. In *Preprints of the AAAI 1992 Spring Symposium Series, Workshop on Computational Considerations in Supporting Incremental Modification and Reuse*, Stanford University, CA, March 1992.

[11] Manuela M. Veloso. *Planning and Learning by Analogical Reasoning.* Springer Verlag, December 1994.