Contingency Planning for an Autonomous Land Vehicle

Theodore A. Linden
Jay Glickaman

Advanced Decision Systems*
201 San Antonio Circle, Suite 286
Mountain View, CA 94040

ABSTRACT: A route planner for an autonomous land vehicle has been developed that deals explicitly with the incompleteness and uncertainties that can be expected in map data. In particular, the planner finds a preferred route taking into account the potential cost of the detours that may be needed if one or more *choke* regions are found to be blocked. In addition, it precomputes and saves the alternative or contingent routes that will be used if any of the potential detour areas are found to be blocked. The planner uses a semantic network representation for terrain which makes it practical for a hierarchical version of an A search algorithm to identify low cost routes where the cost of potential detours is factored into the cost of the route selected. A dynamic programming algorithm is integrated to plan detailed paths for each route segment, with this detailed planning exploiting all of the variations in expected costs which can be estimated based on whatever level of detail is available in the map.

1. INTRODUCTION-REASONING WITH UNCERTAINTY

A route planner for an autonomous land vehicle (ALV), like many other applications of planning technology, must develop a plan using only incomplete and uncertain knowledge about the terrain in which it will operate. While reasoning with uncertain information is a major theme in many AI applications, planning systems offer several additional challenges for reasoning with uncertainty. In particular, planning systems need to develop:

1. Plans that avoid dependencies on events which are unknown or uncertain and are critical to the plan's success. In general, one will prefer robust plans which can succeed even if some of the beliefs on which the plan is based turn out to be wrong.

2. Plans that contain contingency plans to deal with alternative future possibilities. Sometimes the contingency plan is used simply to avoid the cost of computing a new plan during real-time plan execution; however, in other cases the contingency plan may require some actions before the potential problem is even encountered. For example, additional vehicles may need to be moved into position just on the possibility that they will be needed.

3. Plans to obtain additional information that can be used to make better choices during plan execution. This usually involves a tradeoff, and the planner must decide whether the added information is worth the effort required to obtain it.

This paper describes techniques we are using for dealing with incomplete and uncertain map information in robotic route planning applications. Thus far the techniques have been applied to the first two problems above, those of avoiding critical dependencies arid of developing preplanned contingency plans; however, a straightforward extension of the same techniques can be used to evaluate the tradeoffs involved in planning to obtain additional information.

2. BACKGROUND ON ROUTE PLANNING USING MAP DATA

In the eventual applications of the ALV, we can anticipate that maps of some quality will be available. Experience from previous vehicle traversals of the area may also be reflected in a terrain knowledge base. It is important to use this knowledge from maps and from previous experience in generating plans; however, it is also important to realize that:

1. Many obstacles are too small to show up in the map. A map does not contain enough information to determine whether the vehicle can traverse a given path or not.

2. The map records past information and will have some inaccuracies.

Despite these known limitations, most route planners generate a route from the map data that is available and leave it up to the execution time system to replan as necessary. This paper describes some additional things that the advance planning system can do to work around these expected limitations in the map data.

* This work was supported by the Defense Advanced Research Project. Xgency (DARPA) and the U.S. Army Engineer Topographic Laboratories (ETL) under subcontract #GH4-II88I7 from Martin Marietta Denver Aerospard

In typical route planners, the map is divided into regions, and for each region there is some measure of desirability, utility, or cost associated with traversing that region. The regions may be chosen to be uniform square or hexagonal cells; or the regions may be of varying shapes and sizes that conform to the natural features in the map. The cost of traveling through a region can reflect travel time, danger, risk, or other factors that can be derived from the map information. The costs may depend on the robots goals; for example, the cost of traveling on a road would normally be relatively low, but if the robot is trying to avoid being seen, then the cost of areas on or near roads may be high. The effects of weather, lighting conditions, and other factors can also be reflected in the cost calculations. When the map does not have enough information to predict the cost accurately, expected values for the cost are used.

There are a varietj of route phinning algorithms available for finding routes through regions weighted by cost [Keirsey & Mitchell 84, Parodi 86]. For a survey covering many of the related approaches, see [Mitchell & Papadimitriou 85]. We have found that a simple variable sweep dynamic programming algorithm is effective for ALV applications (Linden et al. 86] and can be easily adapted for parallel processing with performance improvements that are nearly linear m the number of processors available.

A limitation of these algorithms is that they do not reason about the uncertainty of the map information except by using expected value computations. Deeper reasoning about the impact of uncertain data can lead to route selections that are more robust and not as dependent on the planner's expectations being fulfilled. For example:

The river crossing problem. Figure 1 illustrates a simple planning problem where there is a barrier such as a river that can probably be crossed at either of two locations. The estimated costs of following the route that crosses the river at B are slightly better than the costs of the route that crosses the river at A. In this example we assume that both crossing points involve the same risk that the crossing point will not be viable. Most route planning algorithms choose to cross the river at B; however, it is probably better to try to cross the river at A because if the vehicle goes to B first, the cost of the possible detour will be much greater.

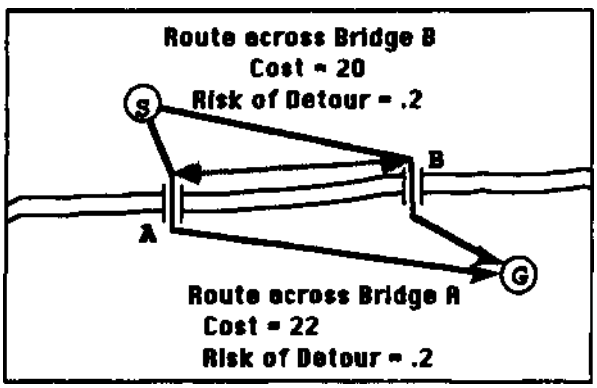

Figure 1: Route Across Bridge B Appears Better, but Detour Costs Make A Better.

The potential dead end. A more serious version of the same problem occurs when a route follows a long valley with only one exit available. An alternative route may be better than the risk of discovering that the entire valley is actually a dead end.

We wanted to develop a practical approach to route planning that would find the better routes in situations such as these. The requirement that the approach be practical for planning routes for typical ALV applications seems to rule out algorithmic approaches that involve calculating the costs of all potential routes and detours. We also considered heuristic approaches that involve examining the map data for all situations similar to the river crossing and potential dead end problems. The difficulty with such approaches is that while heuristics are effective for reasoning about discrete situations, they have a hard time handling problems where all the relationships among the start, goal, and the intermediate blockable points can vary continuously. Furthermore, we were looking for a route planning technique that would apply across a variety of eventual ALV applications. Heuristics for route planning depend on the mission t\pe and on the nature of the terrain involved. For example, it is hard to identify route planning heuristics that apply both to reconnaissance missions where roads are to be avoided and to other tasks where roads are generally to be used as much as possible. Different heuristics also are needed for planning routi m areas where there are a lot of dead ends as opposed to areas with a dense, well-connected road network.

3. OVERVIEW OF THE CONTINGENCY PLANNING SYSTEM

The goal of our work was to find a practical approach toward planning routes for ALV applications which avoids unnecessary dependencies like the one illustrated in Figure 1, which preplans contingent routes that deal with the unavoidable uncertainties, and which can be extended to evaluate tradeoffs involved in gathering additional information.

Our approach is a modified version of a hierarchical A* search (see (Mero 84| for another example) that selects routes first at an abstract level, then in full detail, and then investigates those contingency plans that need to be evaluated to make the full tradeoff between alternative route plans. A preferred route is selected based on the expected cost of following that route where the potential cost of having to execute contingency plans is factored into the cost computations. Contingent routes are saved as part of the selected plan.

An important part of the problem involves finding an abstract, high-level representation of the terrain which is suitable for use in identifying the

most feasible routes. We want this high-level representation to be stable so it will still be useful for any replanning that has to occur while the vehicle is carrying out the plan. For a given vehicle, there is a reasonably stable distinction between terrain areas where the vehicle probably cannot pass (under any weather conditions) and the other normally passable areas of terrain. The cost of moving through a given kind of terrain (which changes with the weather and other factors) is used in the final calculations of the preferred route but is not used in the higher level heuristics.

In addition to passable and impassable regions, we identify a third kind of region that is called a *choke* region. Choke regions are regions where the vehicle can probably travel; however, if it can't, then a relatively large detour around some impassable region will be necessary. Choke regions are relatively narrow traversable regions between two relatively Inrge impassable regions. These regions have the property that, a few obstacles that are too small to appear in the map could make the region untraversable and require not just a minor detour around the small obstacles but a major detour to go around one of the neighboring impassable regions as well. Roads through dense forests, bridges across rivers, and passes through mountains are examples of choke regions.

The three types of regions are calculated from the map data based on both the size of the regions and the features found within them. This process is described in a companion paper (Glicksman & Linden 87|. The results for a region in tne Fulda Gap area of Germany are shown in Figure 2. In this Figure, passable regions are white, impassable regions gray, and choke regions are solid black. The black dots that represent bridges across rivers do not show up well without a color map. Many of the choke regions are long and have complex shapes; for example, some of them are roads within a forest that intersect with other roads.

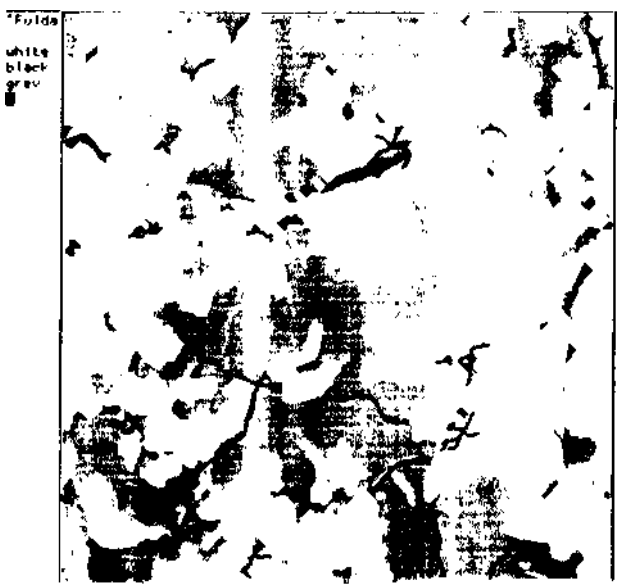

Figure 2: Categorizing Terrain
in the Fulda Gap Region

Once these regions are extracted, we generate their adjacency relationships and compute the size and locations of the boundaries between them. This produces a semantic network where the regions are nodes in the network.

Since choke regions are always narrow, their boundaries with passable regions are short. This is important since the problem of deciding where to cross the boundary between two adjacent regions is one of the harder problems when planning shortest paths through region-oriented representations of terrain. The decision about where to cross one boundary depends on where the preceding and following boundaries are crossed, and there is no good way to get an admissible estimate of the path cost until these interdependent decisions about boundary crossings are made. By using the concept of choke regions, all boundaries are guaranteed to be short enough so the midpoints of the boundaries can be used with only minimal loss of accuracy.

The first stage in the hierarchical A* search uses the semantic network representation of the terrain and an admissible estimate for the cost of traversing the individual route segments within each region. Once a route through this high level representation is found, a detailed calculation is done, using a grid-based representation of weighed traversal costs, to find actual paths for each route segment and a real cost for that segment. This detailed calculation is only done for those route segments which are part of candidate routes.

Once t lie leading candidate route has been calculated with full detail on all of its route segments, then the exploration is continued in order to find the contingent paths that would be followed if any of the choke regions used in the primary path are found to be blocked. The cost of following these detour paths is then weighted by the probability that the detour will be necessary, and this additional cost is factored into the estimated cost of traveling this route. Note that the estimated cost of following a route increases as the contingencies are explored. As this happens, the A* algorithm that is guiding the search will focus its attention on alternative routes which will be explored until their estimated costs rise to be higher than the cost of some other route.

4. ABSTRACT PLAN REPRESENTATION

At the highest level of abstraction, route plans are represented in terms of the sequence of choke regions that will be traversed. We use lists to represent these abstract plans. For example,

(ST A B C GL)

represents a plan to travel from the start (ST) through choke regions A, B_f and C in that order and then on to the goal (GL) without passing through any other choke regions. Each of the variables is really a pointer to a region data structure. The start and goal are artificially generated "regions" that consist of a point in a passable region. All the other variables

represent pointers to choke regions. This representation omits the passable regions that must be traversed between every pair of choke regions in the list because they can be unambiguously determined from the regions in the plan.

To represent the contingency plans that would be executed if a choke region is blocked, we embed a sublist immediately after the choke region in question. For example,

(ST A (X Y GL) B (Z W R GL) C (B Z W R GL) GL)

describes a plan that contains contingent routes that will be followed if region A, B, or C is unavailable. In this example, if choke region C is discovered to be blocked, the path taken will be

(ST A B C B Z W R GL)

In other words, upon reaching region C and discovering that it is blocked, the plan is to backtrack through B and then take the alternate route through Z, W, and R (and the passable regions between them) to the goal. Deeper nestings of plans can be used to represent routes that deal with multiple blockages of choke regions.

It is possible that at some level there will be no more alternates available. This is signified in the plan representation by the word BLCK (for blocked). For example,

(ST A (BLCK) B (I (U V GL) J (W Z GL) GL) GL)

indicates that if A is unavailable then one cannot reach the goal.

The planning algorithm is parameterized so that one can specify the number of levels of contingencies to explore.

5. THE CONTINGENCY PLANNING ALGORITHM

The implementation of the contingency planning algorithm is described in this section. The algorithm is an extension of a hierarchical A*. The cost function is described in the next section. A flow chart of the control flow for the algorithm is shown in Figure 3.

The start and goal positions are inputs; and an additional parameter, max-level, determines the number of simultaneous blockages of choke regions that the algorithm will consider while evaluating the cost of detours. When max-level is 0, no contingency planning is done, and the algorithm reduces to a hierarchical A*.

As a first step, a bi-connectedness algorithm (a modification of (Sedgewick 83| p. 392) is run on the graph of linked choke regions to determine if there is a path from the start to the goal and whether there are any articulation points which must be used in every path from start to goal. This algorithm is O(n) in the number of links between choke regions, and its use here forestalls more expensive searching for contingent routes when that search is doomed to fail.

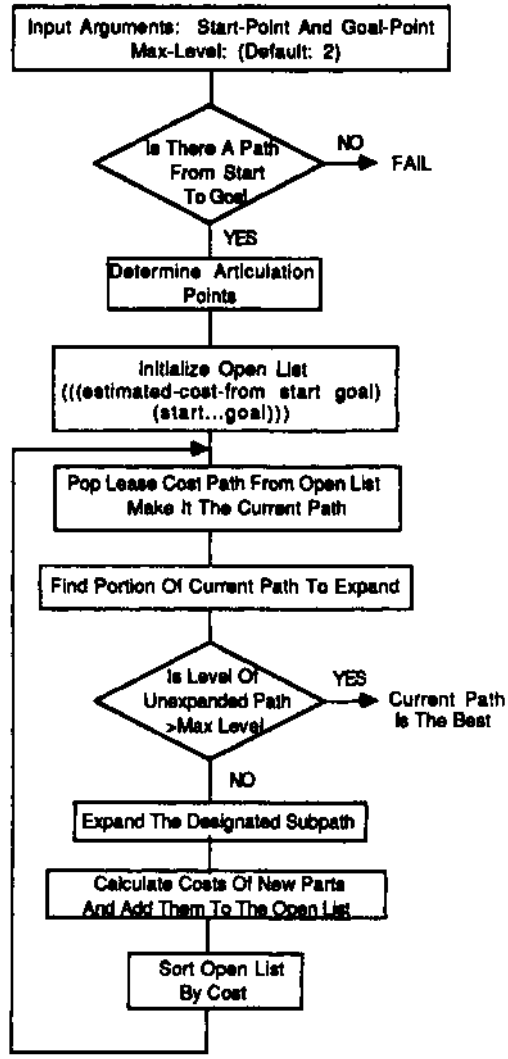

Figure 3: The Contingency Planning Algorithm

Potential plans are placed on an open list along with their cost. The least cost plan on the list is the current "best" candidate in the hierarchical A* algorithm. When a complete plan with "real" costs reaches the top of the list, then it is returned as the result. Otherwise, it is expanded one step closer to the goal. The cost of each newly generated plan is determined so it can be placed in the appropriate spot on the open list. For partial plans, cost estimates are used. When a path is "complete"[1] (i.e. represents an abstract plan through a sequence of choke regions all the way to the goal), then a path for each route segment between choke regions is computed using a dynamic programming algorithm running over a cost matrix that records the traversal cost for each small area of terrain. The cost of traveling through each of the choke regions involved in this route is also computed. While these detailed traversal cost.*are themselves estimates based on the map and other collate™I

(lata that is available, in this paper we refer to them as the "real" cost because they are the most accurate cost data available to the algorithm and are more accurate than estimates based on straight line distances.

In the following, ... designates an unexplored portion of the partial plan to be expanded. A subplan can occur at any level in the complete plan. The expansion of the plans considers two different cases:

Case i: There is a subplan is of the form (W A D ... GL).

In this case, all the nodes that can be reached from D and are not already in the subplan at this level are added to the subplan and the results are added to the open list. Furthermore, if the subplan is not already at max-level in nesting depth, then contingent paths are added after D. For example, the above path becomes:

(W A D (... GL) X ... GL)
(W A D (... GL) Y ... GL)

If the goal can be reached directly from D then

(W A D (... GL) GL)

is also added to the open list. Removing the three dots to form a complete path at a given level also causes a switch in the way the costs are determined: from estimates to "real" costs as calculated by dynamic programming.

If there were no new nodes that could be reached from D then this branch of the search tree might represent a dead end. In that case, the sublist is replaced with (BLCK) as in

(X (BLCK) GL)

This would arise in the case where X was not an articulation point but some combination of blocked nodes prevented a path from being found from the start to the goal along that alternative.

Case ii. The subplan is of the form (W A (... GL) B GL)

This case arises when moving down one level in the contingency hierarchy; that is, when it is the first exploration of the contingent path to be followed if A is unavailable. Call A the current choke region.

If A is an articulation point, the subplan (... GL) is replaced by (BLCK).

If the route is not known to be blocked, then the following procedure is followed: back up past all the unavailable regions to the first available region (W in the example). Then find all the nodes that can be reached by both the current choke region (A) and the available region (W) but that are not otherwise unavailable and those are the regions that should be added to the open list. For example:

(W A (X ... GL) B GL)
(W A (Y ... GL) B GL)
(W A (W ... GL) B GL)

When a new choke region is added to the plan while it is being expanded, if the nesting level of that element is lower than max-level, then an incomplete alternative path is also added; thus, the expansions of the plans listed above may actually have the form:

(W A (X (... GL) ... GL) B GL)
(W A (Y (... GL) ... GL) B GL)
(W A (W (... GL) ... GL) B GL)

8. COST FUNCTION

The calculation of estimated and real costs for any partial plan uses the following inputs (some of which are computed on demand):

1. For each choke region A, the probability P_A that the region will be traversable.
2. For pairs of choke regions A and B that are connected by a single traversable region, both the estimated cost E_{AB} and the real cost C_{AB} of traveling from one choke region to the other.
3. For each choke region A, an estimate of the cost of traveling from that choke region to the goal E_{AC}.
4. For each choke region A, an estimated cost E_A and a real cost C_A of traveling through it.

While most of these values are not calculated until they are needed, once a value is determined, it is stored in a table because it is likely that it will be required repeatedly in both the current plan (as it is expanded) as well as others.

For the result to be admissible, the estimated costs must be less than or equal to the real costs. In the implementation of this algorithm, the estimated cost is the minimum cost of traversing any particular grid cell in the region multiplied by the straight-line distance between choke regions (for E_{AB}) or by the straight-line distance between the two ends of a choke region (for E_A). In the case of the cost between a choke region and the goal where the choke region is not adjacent to the region containing the goal (E_{AC}), the minimum cost for all cells is used. The real cost (plus the real path) for each segment is determined by using a dynamic programming algorithm which computes the best path for that segment. The estimated cost will always be smaller than the real cost.

When the alternative paths are factored in, the cost is effectively split in two: the cost of the path that would be taken if the choke region is traversable is multiplied by P_A. Similarly, the cost of the path that would be taken when a detour is necessary is multiplied by $(1 - P_A)$, the probability that a detour is required.

Given that the estimated and real costs for each route segment can be calculated when needed, the cost of a plan can be defined by induction following the process by which the plan was generated as described in Section 5. The key steps of this induction are:

1. If the previous plan was (W A D ... GL) and the current plan is (W A D (... GL) X ... GL), then replace the cost E_{DG} with $P_D * (E_{DX} + E_{XG}) + (1 - P_D) * E_{DG}$

2. If the new subplan has the form (W A D GL) with the three dots removed, then recalculate the cost of this subplan using real costs rather than estimates for every segment along this path.

3. If the new subplan has the form (W A (X ... GL) B GL) and the previous plan had the form (W A (... GL) B GL) then replace E_{AC} with $E_{AX} + E_{XG}$.

4. If the new plan has the form (W A (BLCK) B GL) then don't change the previous cost estimate. Note that any change in cost in this situation would eventually affect all open plans equally, and thus the assignment of a cost for this case ultimately will not matter. If there is no alternative path when A (and possibly other nodes) is blocked, then every path expansion will eventually try A under the same set of blockage assumptions and will end up encountering the same cost when it also assumes that A may be blocked.

The actual implementation does not save the separate components of the plan's cost, but rather recalculates the cost using an algorithm equivalent to this inductive definition.

7. RESULTS

The algorithm for contingency planning has been implemented on a Symbolics Lisp machine and run using map data for the Fulda Gap area in Germany as well as map data for the Martin Marietta ALV test site near Denver, Colorado.

Figure 4 shows an example of the output generated during the execution of the contingency planner. The final route (on a black and white device) is displayed with the "primary" path as a solid line (with solid arrowheads added to make it more visible) and all the contingent paths as dotted lines (with hollow arrowheads added). The plan generated in this example is

(ST 218 (117 140 248 340 GL) 296 (288 340 GL) 298 (340 GL) GL)

where the numbered choke regions are actually the print names of internal objects. This route is interesting because if the first detour region (218) is unavailable, one has to backup past the original starting point and move away from the goal for some time before making progress towards the goal.

Figures 5 and 6 show examples where the contingency planning makes a difference in the choice of routes. In both cases the main route found when no contingencies are determined (in Figures 5a and 6a) is not considered the "best" route when a level of contingency planning is used.

Figure 4: A Contingency Planning Example

8. EXECUTION TIME

Table 1 shows some timing results for the contingency planning algorithm as read from the real-time clock on a Symbolic 3670 Lisp machine with all intermediate output turned off. The ephemeral garbage collector was turned on. Two parameters were manipulated during the timing experiments: the amount of dynamic programming to generate detailed plans with "rear" costs was varied as well as the number of levels of contingency planning. No DP means that the dynamic programming was turned off and "Choke only" means that dynamic programming was only used within choke regions. The timing results show that the vast majority of the computation time goes into the dynamic programming computation to develop correct detailed routes. We are using a special form of dynamic programming which can exploit parallel processing effectively (nearly linear speedup for up to a few hundred processors) [Linden et al. 85). Unfortunately, the current timings are on a serial processor that is not even designed for numeric computation. We have evidence that running the dynamic programming on an appropriate parallel processor would speed up that part of these computations by at least two orders of magnitude.

These timings were taken while planning the first two-thirds of the plan illustrated in Figure 4. The first set or timings did not use contingency planning at all and the final plan was

(ST 218 296 GL)

The second set of timings generated one level of contingencies and the final plan was

(ST 218 (117 140 248 GL) 296 (288 GL) GL)

Table 1: Timing Results

DP Use	Contingency Level	Time (seconds)
None	0	7.6
Choke only	0	31.2
All	0	1801.0
None	1	7.7
Choke only	1	193.2
All	1	2855.6

g. LIMITATIONS AND FUTURE WORK

The major bottleneck in the execution time of this algorithm is the calculation of detailed route segments and their costs. Some effort was spent in making the dynamic programming for this run efficiently; but, in addition to parallelism in the hardware, there are other steps that might be taken. When it is known that many routes are going to be planned through the same terrain, the calculation of the individual path segments can be done once for all the combinations of adjacent choke regions and saved for future use. Even if some of the path segments have to be recalculated because the weather or other factors have changed, these precomputed costs can be constructed to be admissible and they are likely to be much better estimates than an estimate based on straight line distances.

Figure 5: Where Contingency Planning Makes a Difference
a. No Contingencies; b. One Level of Contingencies

Figure 6: Where Contingency Planning again Makes a Difference
a. No Contingencies; b. One Level of Contingencies

Choke regions frequently connect three or more passable regions or they connect to the same passable region in two or more places. The current implementation does not always generate the right route in the vicinity of such choke regions. One solution (that has not been implemented) would divide choke regions that have multiple exits into separate choke regions which have two exits each and are connected to each other by "passable regions" that are just a single point.

While expanding paths towards the goal, it is possible for the same contingent paths to be generated several times. Moreover, alternate "losing" paths will be regenerated and re-evaluated while the paths are again explored. If branches of the search tree are stored as they are completed, then they can often be re-used in almost "equivalent" situations and thus reduce the amount of search. This becomes more important with longer paths and more levels of contingencies because the combinatorics of the problem increase dramatically.

With a couple of minor enhancements, the present algorithm could be used to evaluate the use of potential shortcuts. Shortcuts go through areas which are probably not traversable, but might be. Thus ."shortcut regions" are the same as choke regions except that the probability of successfully traversing them is less than one half. The current algorithm makes no assumptions about the probability of successfully traversing a choke region—except that the cut-off in terms of the depth of searching for contingency plans should then be handled in terms of the probability of actually executing a path rather than in terms of a fixed number of contingency levels. This option is implemented but has not been well tested.

The present algorithm could be used to evaluate the tradeoffs involved in planning to obtain additional information as opposed to forging ahead without it. For example, a future robot with good long distance vision might need to decide whether it is worth the effort to climb a hill to see whether a bridge is still usable-as opposed to traveling directly to the bridge.

The current implementation uses a linear evaluation function and considers the effect of detours on the expected value of the cost. By extending it to use a non-linear evaluation function, we could deal with more complex goals; for example, where it is important to arrive before some fixed time. The need to consider the effect of potential detours becomes even more significant in such cases.

10. CONCLUSIONS

We have developed a route planning algorithm that generates good routes despite unknowns and uncertainties in the map data being used; in particular, the algorithm identifies choke points and takes the cost of potential detours into account while it is selecting the preferred route. It also generates contingent routes that will be used if choke points turn out to be impassable, and it can be extended to generate plans that obtain additional information that is then used to extend the plan.

We believe that this algorithm is practical for use in mobile robots. When generating complete routes with one level of contingencies, the 10 to 60 minute execution times for the algorithm are slow enough to make debugging and testing a painful process, but execution times of that order of magnitude are reasonable for pre-mission planning for many mobile robot applications. Furthermore, most of the time is spent calculating detailed routes, and that calculation can be improved by a couple orders of magnitude either by using parallel hardware or by precomputing the detailed route segments and re-using them in subsequent route planning sessions.

11. REFERENCES

[Glicksman & Linden 87 Jay Glicksman and Theodore A. Linden, "Terrain Reasoning to Support Contingent Path Planning," forthcoming.

[Keirsey & Mitchell 84] D. M. Keirsey and J. S. B. Mitchell, "Planning Strategic Paths through Variable Terrain Data," Proc. SPIE Applications of Artificial Intelligence, 1984.

[Linden et al. 86] Theodore A. Linden, James P. Marsh, and Doreen L. Dove, "Architecture and Early Experience with Planning for the ALV," Proc. 1986 IEEE International Conf. on Robotics and Automation, April, 1986, pp. 2035-2042.

[Mero 84) L. Mero, "A Heuristic Search Algorithm with Modifiable Estimate," Artificial Intelligence 23, 1984, pp. 13-27.

[Mitchell & Papadimitriou 85] Joseph S. B. Mitchell and Christos H. Papadimitriou, "Planning Shortest Paths," Proc. SIAM Conf. on Geometric Modeling and Robotics, Albany, NT, July, 1985.

[Parodi 84] A. M. Parodi, "A Route Planning System for an Autonomous Vehicle," Proc. 1st. IEEE Conf. on AI Applications, Denver, Dec. 1984, pp. 79-84.

[Sedgewick 83] R. Sedgewick, Algorithms, Addison-Wesley, Reading, Mass., 1983.