TACTICAL ROUTE PLANNING: NEW ALGORITHMS FOR DECOMPOSING THE MAP

J. R. BENTON
Modeling and Simulation Division
U. S. Army Topographic Engineering Center
7701 Telegraph Road
Alexandria, VA 22315-3864

S. S. IYENGAR, W. DENG, N. BRENER
Department of Computer Science &
Robotics Research Laboratory
Louisiana State University
Baton Rouge, LA 70803-4020

V.S. SUBRAHMANIAN
Department of Computer Science &
Institute for Advanced Computer Studies
University of Maryland
College Park, Maryland 20742

Received 17 April 1995
Revised 10 January 1996

ABSTRACT

This paper defines a new approach and investigates a fundamental problem in route planners. This capability is important for robotic vehicles(Martian Rovers, etc.) and for planning off-road military maneuvers. The emphasis throughout this paper will be on the design and analysis and hierarchical implementation of our route planner. This work was motivated by anticipation of the need to search a grid of a trillion points for optimum routes. This cannot be done simply by scaling upward from the algorithms used to search a grid of 10,000 points. Algorithms sufficient for the small grid are totally inadequate for the large grid. Soon, the challenge will be to compute off-road routes more than 100 km long and with a one or two-meter grid. Previous efforts are reviewed and the data structures, decomposition methods and search algorithms are analyzed and limitations are discussed. A detailed discussion of a hierarchical implementation is provided and the experimental results are analyzed.

Keywords: Route planning, heirarchical planning, A* algorithm, robotic vehicles.

1 Introduction

Route planning is an old discipline within computer science and has been applied in such diverse fields as: determining optimum routing of electrical paths on a printed circuit board; optimizing helicopter flight paths under constraints of enemy threats, weather and fuel consumption; and optimizing the traversal of a vehicle across terrain. Terrain traversal, in turn, can be subdivided into on-road, off-road or a combination of the two. This last area of off/on-road route planning is the subject of this paper. What is common to all route planning is the need for efficient search. What is particularly challenging for terrain traversal is the evolving need to plan the coordinated movements of multiple vehicles with differing starting points and destinations subject to the constraints of the modern battlefield. An elevation grid with as many as a trillion intersections may be required to represent this battlefield. The route planning must take into account not only terrain factors but also minimize the risk of being destroyed or damaged by enemy action. Additionally, the planning must be fast enough to support real-time simulation of vehicles on terrain. New approaches to representing data and searching the data space are required. This paper reviews previous work, describes a new approach in detail and discusses ideas for future research.

A motorist needs a good road map that contains all the roads across which he might want to travel. Similarly, in planning our terrain traversal, we need the equivalent of that road map. However, our vehicle will be traveling off-road as well as on road and it is impossible to know ahead of time what either the starting point or destination will be. Therefore, we need an estimate of possible speed everywhere on the map. Ideally, we would like to know how fast we can go in the up-slope, down-slope and cross-slope directions.

Terrain data can be as simple as an array of elevations (which provides only a limited means to estimate mobility) or as complex as an elevation array combined with digital map overlays of slope, soil, vegetation, drainage, obstacles, transportation (roads, etc.) and the quantity of recent rain. We can use the NATO Reference Mobility Model software [1] to compute the allowable speed at each grid point for a particular type of vehicle. Considerable manual efforts are required to produce the overlays and this type of data is available only in a few regions of the world. The usefulness of even this data is limited by the low resolution of the data; most of it is 100 meters with a very limited amount of 30-meter data. A lot of obstacles can be hidden in a 100-meter grid cell. These factors have all served to limit the actual use of automated off-road route planning.

What's new is the expectation that in the near future, we will have available elevation grid data at resolutions of five meters or better. This data will be primarily from Interferometric Synthetic Aperture Radar (IF-SAR) systems. Scientists at

JPL [2] have demonstrated that relative accuracy of better than five meter can be obtained.

When elevation data is only available at 30-meter or 100-meter spacing, it is imperative to have auxiliary data (i.e. overlays) to determine if a the grid cell is traversable. In contrast, with a one meter grid, individual trees and boulders will be discernible in the elevation data as obstacles. Knowledge of the soil type can not be easily discerned from radar data. However, soil types change relatively slowly and given knowledge of the geographic province, recent weather data, slope and drainage, inferences can be drawn as to how much the soil conditions will impede mobility for any specific vehicle type.

Automated route planning will be a key element of almost any automated terrain analysis system that is a component of a military Command and Control System. However, instead of simply using the traversal cost factors from the mobility model, a new cost surface must be constructed in which the total cost is a linear combination of the traversal cost and a *threat* cost that is assigned to traversing areas adjacent to some threat. A threat could be proximity to enemy weapons, visibility from enemy observation sites or terrain factors which would make it difficult to defend against an enemy attack. The threats are often modeled by a cone centered on the threat so that a selected path would be allowed to cross over the outer perimeter of a cone or alternatively a cylinder may be used which will totally interdict travel throughout the radius of the cylinder. Other more complex risks such as visibility from enemy observation posts or from the enemies side of a battlefront can also be included in the cost. Examples of military applications include: deployment of a network of air defense missile batteries [3, 4], generation of military Avenues of Approach (AoA) which are used to engage an enemy force, determination of likely minefield sites by analysis of choke points that limit mobility, route planning of multiple column of tanks, automated control of mine laying or mine sweeping robotic vehicles and robotic reconnaissance vehicles. Nonmilitary application include: (1)interplanetary exploration - In addition to their visual systems, the Martian or Lunar rover will rely on knowledge of the topography to avoid dangerous area or dead end paths and (2) disaster relief operations: After a flood or serious earthquake, well known routes will no longer be available. IF-SAR data will provide an opportunity to update databases rapidly and to determine off-road paths around blocked roads.

There have been recent advances in deductive database technology and mediator frameworks[5] for easily integrating both heterogeneous sources of data and software systems into a coherent whole. Use of HERMES [6, 7, 8] (HEterogeneous Reasoning and Mediator Environment System) will allow the answering of queries that require the interrogation of multiple databases in order to determine the start and destination parameters for the route planner. HERMES was recently integrated with a grid-level planner developed at TEC and complex queries that required accessing both a relational database and the route planner were successfully answered.

In summary, the elements required for significant improvement in off/on-road route planning are coalescing and practical route planners capable of planning high resolution paths across a distance of 100 kilometers or more will be become available. Solutions are being found for collecting high resolution terrain data, navigational systems are achieving the accuracy to tie the data to the world geoid, hierarchical approaches to route planning will provide the needed computational power, and expert systems and deductive databases will provide the intelligence to integrate the route planning into higher-level systems.

Sec. 2 reviews some of the algorithms developed for off/on-road route planning. Sec. 3 is a detailed description of a prototype system, Sec. 4.1 discusses a cooperative approach to planning multiple routes, Section 4.2 describes an intelligent terrain reasoning system and Sec. 5 contains concluding remarks.

2 Previous Work

This section describes several alternative search algorithms and other techniques used to simplify the search. This review is not intended to be exhaustive, but rather to present representative trends in path planning research. Furthermore, algorithms developed for the special binary case of *go* or *no-go* are excluded.

Dijkstra's Shortest Path Algorithm and the A* algorithm have been the most popular approaches to off-road route planning. Most of these planners operate at the level of the elevation grid, often called the pixel level. Planners normally use eight-neighbor connectedness in selecting the next pixel on the path. The angular quantization is then forty-five degrees. We can easily see the effect of this quantization if we consider an eight by eight chess board. If we assume that the traversal cost between adjacent squares is 1.0 in the cardinal directions and $\sqrt{2}$ along the diagonals then the optimum path from one corner of the board to the opposite corner is a straight line along the diagonal. In this case the true path is at an angle corresponding to the angular quantization and no error is introduced by the quantization. If two checkerboards are placed side by side, then the path from the lower left corner of the left board to the upper right corner of right board requires eight diagonal moves as before, but also eight horizontal moves. The horizontal and diagonal moves can be distributed arbitrarily and the distance will remain unchanged. Remedies for correcting this digitalization error are discussed in a paper by Mitchell [9]. The correction is based on whether the current pixel along a path represents a change in direction or is a continuation along the same direction. The cost is reduced if there is a change in direction. This approach does not globally optimize and does not result in only one optimum path, but it does prune those paths whose smooth path distance is significantly greater than the smooth path distance of the optimum path.

It is interesting to contrast the A* algorithm to the potential field method [10, 11]. This approach is based on the metaphor of an object always moving downhill

seeking the point of minimum potential (lowest point). For our motion planner, the potential is composed of two factors. The first is a quadratic surface with its minimum at the goal point. This first surface is superimposed with a potential field which at each grid cell is proportional to the difficulty of traversing that cell. The combined potential provides an attraction toward the goal and a repulsion from obstacles. However, if this method is not combined with another method that provides for backtracking, the robot is likely to be trapped in a local minima. The quadratic surface is the equivalent of the heuristic function of A*, and within the context of A*, is an admissible function.

The use of A* algorithm to compute off-road paths is not limited to searching a grid. Three approaches have been developed in which the map of grid cells is decomposed into a graph. All of them first convert the map into regions of *go* and *no-go*. The *no-go* areas may be considered obstacles and are represented as polygons.

- *Visibility Diagram:* Nilsson [12] developed the concept of generating a graph as follows: For each obstacle polygon, visit each vertex and draw an arc to every other vertex that is *visible* and for which there is not already a connecting arc. Two vertices are *visibile* to each other only if the traversing arc does not cross the interior of any polygon. Note that if a polygon is concave, at least two vertices of the polygon will be mutually visible and will have arcs connecting them. The start and destination nodes are inserted and for each of the nodes, arcs are drawn to all visible vertices. If the nodes are mutually visible, than an arc will be drwn directly between the nodes. The A* algorithm uses the resulting graph to compute the optimum path. A variant of this approach is to use polygonally-defined regions of the areas which are not visible from specified observation points. The ModSAF (Modular Semi-Automated Forces) simulation system has a route planner [13] which chooses routes that minimize exposure from these observation points.
- Another way to consider the map is to think of obstacles in the map which force a decision to go to the left or to the right to get past the obstacle. The decision point corresponds to a node in a graph with the arcs representing the paths around the obstacles with one obstacle contained in each polygon of the resulting graph. Two alternatives for generating the obstacle graph are Voronoi diagrams and line-thinned skeletons (which are closely related to medial axis transformations [14].) The Voronoi diagrams generate paths which are equidistant from *no-go* areas. The medial axis graph is in many ways similar to the Voronoi diagram. An important difference is that if along the map boundary, the region is *go*, then an adjacent interior obstacle will be enclosed in a polygon of the skeleton. In the corresponding Voronoi, the obstacle will not be enclosed. The desired behavior is to have the obstacle inclosed since this provides correspondence to the map which does show a path along the map boundary. The Voronoi approach is used in two systems described below. The line-thinning approach was implemented by Benton [3] in 1988 and is described in greater detail in Sec. 3.1

Military columns of armor normally travel along Mobility Corridors, off-road routes that meet the requirements of some minimum width of the corridor and with ground conditions that the specified number of vehicles will be able to traverse before the route becomes untraversable. The width requirement is greater for a tank company than for a tank platoon. Powell[15] used computational geometry techniques to extract the Mobility Corridors from a traversability cost map. Voronoi and their complementary Thiessen Triangulation were used to extract the military corridors.

Marti [16] at RAND Corporation developed a combined on-road/off-road route planning system that was closely integrated with a geographic information system and a simulation system. Routes can be planned for either single columns or multiple columns. For multiple columns, the planner keeps track of the temporal location of each column and insures they will not occupy the same space at the same time. The road network used in their work contained over 9500 road intersections and 12,800 arcs between them. Both planners use a breadth-first search. The off-road planner uses a variable grid size which is determined by the local terrain. The costs are derived as individual grid cells are searched since both grid size and location are subject to change. In areas where there are obstacles that can impede movement, Voronoi diagrams are used to derive the medial path between the obstacles. These paths are combined into a graph that can be used by the route planner. This system differs from the others in that it does not use a precomputed cost surface and it has elements of a hierarchical system.

Mitchell, Peyton and Kiersey [17] incorporated a route planner with multi-level hierarchical control into a robot visual simulation system. The bottom layer of the hierarchy is a vision-based route planning system with the vision supplied by simulating an acoustic ranging device. The system permits planning of routes from an arbitrary starting point to an arbitrary end point. If the starting point is off-road, the planner uses A* to compute a path to an adjacent road. A graph-based A* planner then computes the optimum path along the road network to the point on the road nearest the destination. The off-road planner then completes the path to the destination. The off-road planner also performs the correction for angular quantization error described above. If the simulated vehicle senses an obstacle while traversing a road, then a reflex planner attempts to go not more than some set distance off-road to get around the obstacle. If the reflex planner fails to find a detour, then the higher-level planner is called to replan the route from the current location to the destination. This replanning may require backtracking along the road just traversed. The obstacle encountered on the road can be a fallen tree, shell crater, etc. The user of the simulation can interactively lift an obstacle onto the road. This change is automatically incorporated in the database.

Kreitzberg [18] at JPL has developed the Tactical Movement Analyzer (TMA). The system uses a combination of digitized maps, satellite images, vehicle type and weather data to compute the traversal time across a grid cell. TMA can compute optimum paths that combine both on-road and off-road mobility, and with weather conditions used to modify the grid cost factors. The smallest grid size used is

approximately 0.5 km. Kreitzberg uses the concept of a signal propagating from the starting point and uses the traversal time at each cell in the array to determine the time at which the signal arrives at neighboring cells. The earliest time for arrival at a cell is saved. All eight neighbors of a cell on the queue are examined. The advantage of this algorithm is that it is not necessary to make additional calculations to determine the time at which a vehicle arrives at a given cell. The concomitant disadvantage is that traversal is the only factor in computing the optimum path; risk factors do not affect the choice of the optimum path.

Another approach has been to develop algorithms which are parallelizable. Furthermore, if the algorithm depends only on the nearest neighbors, then it can be made to run on a very simple data parallel architecture known as the Cellular Automata (CA). Each cell is a processor which communicates only with its nearest neighbors and can perform simple arithmetic or boolean operations on the data stored in the cell and its adjacent neighbors. The results of the operation are stored in the cell. Stiles and Glickstein [19] developed a Parallelizable Route Planner (PRP) algorithm which is well adapted to running on a cellular automata. For each cycle of the automata, the wavefront of the search expands to the adjacent unexplored nodes. Thus the minimum number of iterations required to determine a path is the length of the path measured in number of cells traversed. Stiles and Glickstein implemented the CA on a CM2 Connection Machine with 65536 processing elements. Their specific application was for helicopter route planning, but their algorithm can obviously be adapted to ground-based vehicles.

Other researchers have chosen to decompose the map into regions that are defined by having a constant traversability across the region. The advantage of this approach is that the number of regions will, in general, be far fewer than the number of grid cells since the regions always consist of one of more grid cells. The disadvantages include difficulty in defining the center of the region and the computation difficulties in determining the optimum path between two adjacent cells. Richbourg [20] and Mitchell [9] have made use of an analogy with the Snell's Law used in optics. This law specifies the path of a ray when it crosses from a region with one index of refraction to another region with a differing index of refraction. The ray traced by a path conforming to Snell's Law is a minimal traversal-time path. Thus by substituting the reciprocal of the maximum traversal speeds for the corresponding optical indexes of refraction in Snell's Law, one is guaranteed (with some exceptions[a]) that a path generated by this analogous Snell's Law will be a minimal traversal-time path. However, this procedure provides only local optimization between adjacent nodes. A simultaneous solution for the optimality requirement at all the region edges leads to a polynomial of degree exponential to the number of edges along the path [9]. The optimum region-to-region path can be obtained by using either Dijkstra's Continuous Algorithm (DCA) developed by Mitchell [9] or an A* approach developed by Richbourg [20]. For the DCA, polygonal regions

[a] However, the optical Snell's Law must be modified for the case of critical reflection in which the angle is sufficiently large that there is total internal reflection of the light ray. In this case the optimum path will travel along the edge and exit later.

must be decomposed down to triangles while Richbourg's algorithm allows arbitrarily shaped polygons. Both algorithms use iteration to refine an initial approximate path to the final optimized path which obeys Snell's Law at all boundary intersections. Mitchell optimized for worst case performance while Richbourg designed his algorithm to optimize average-case performance. This was done by extensive use of pruning and heuristics to keep the search space small. The DCA has time complexity of the seventh power of the number of edges in the map. The worst case performance of DCA is much better than that of Richbourg's A* algorithm but the average-case performance of the A* is better than that of the DCA. In summary, Richbourg states that the "two algorithms rely on common precepts but have fundamentally different capabilities and operational characteristics" [20].

3 Prototype Hierarchical System

The route planners described in Sec. 2 use many innovative concepts and have a wide variety of capabilities. Several of these systems are within some sense of the word hierarchical. If, however, hierarchical is defined to require that search methods be applied at two levels of resolutions, then none of these systems can be called hierarchical. None of them can be easily extended to provide high resolution planning (one meter) over distances of one hundred kilometers or more. The prototype system described in this section does meet this strict definition of hierarchy. In addition, Sec. 4.2 describes the use of a deductive database and a mediator to ease the integration of our route planner into larger systems and also provides a high level logical programming language for framing queries which include calls to both relational data bases and the route planner.

This new system is currently being developed in a joint effort by the U.S. Army Topographic Engineering Center (TEC) and the Computer Science Department of Louisiana State University (LSU). The new system is called Predictive Intelligence Military Tactical Analysis System (PIMTAS). It integrates the Hierarchic Route Planner [3, 4] previously developed at TEC with software developed at LSU. PIMTAS will use an LSU-developed event-driven version [21] of the CLIPS [22] expert system shell and the PIMTAS grid-level route planner integrates the best features of the LSU and TEC planners. The aerial planning capability of the LSU planner is included in PIMTAS.

The primary rationale for developing a hierarchical route planner was to avoid the intensive computation of a grid-level route planner in computing a path with a length of over a thousand pixels. An additional requirement was that no potential routes be discarded simply because the path is too narrow to show up at the higher level of the hierarchy. This last requirement precludes the possibility of initially using a coarse grid and successively refining the path found at the coarse level with a finer resolution grid. The HRP and its enhancements under PIMTAS will be described in the following subsection with a concluding subsection describing the use of an event-driven expert system shell that will provide the overall intelligence

of PIMTAS. Several examples will be included showing how PIMTAS will provide intelligent and adaptive control for rapid, real time responses to unexpected, real time events.

3.1 A Hierarchical approach to route planning

In Sec. 2, it was shown how the map plane can be decomposed into regions based on aggregating pixels with similar traversability characteristics. Another way to consider the map is to think of obstacles in the map which force a decision to go to the left or to the right to get past the obstacle. The decision point corresponds to a node in a graph with the arcs representing the paths around the obstacles with one obstacle contained in each polygon of the resulting graph. Line thinning is the method used to generate the skeleton.

The skeleton of the *go-to* areas acts as the link between the two levels of the hierarchy. The skeleton is first converted into a graph structure with the nodes representing decision points and the arcs representing the alternative paths that can be taken. Arcs are an artifact of the thinning process and do not take into account the cost of traversing a given pixel. Therefore, a grid-level route planner is required to determine the actual paths between the nodes and to assign a cost to the traversal. A graph-level planner does long distance planning across an expanse of many nodes. However, the nodes themselves are an artifact of the thinning process and the final path should not be constrained by the locations of the nodes. Therefore the final step is a relaxation process which removes the constraint imposed by the nodes of the skeleton. This is accomplished by defining new nodes half way along the path between the original nodes. The graph-level planner then computes the path between the new nodes. Typically, the relaxation step improves traversal time by about ten percent. Roads can be given a preference for use by the HRP by specifying a higher speed category for roads then for any off-road region. The grid-level planner will then automatically select a road for travel whenever travel on the road results in a lower overall cost. In summary, the Hierarchic Route Planner makes use of the full resolution of the data in making the skeleton and the combination of grid-level and graph-level plans provides precise and accurate planning across long distances.

The advantage in the hierarchy used in the HRP is that the reduction in the number of nodes from the grid level to the graph level is on the order 1000 to 1. If the spacing between nodes in the graph averages 50, the length of the search path is 30 nodes deep, and a cartesian distance between the start and goal is 1000 pixels, then the number of grid-level pixels that would need to be expanded would be on the order of one million. In actual practice, far fewer than 1000 nodes would be expanded in the graph search. Thus there is a 1000 to one increase in speed in this example. The disadvantage of the HRP is that in a pre-processing step, all the nodes of the graph which are directly connected must have their connecting path and associated cost computed using the grid-level planner. This is a one-time cost and thereafter long distance routes can be quickly computed.

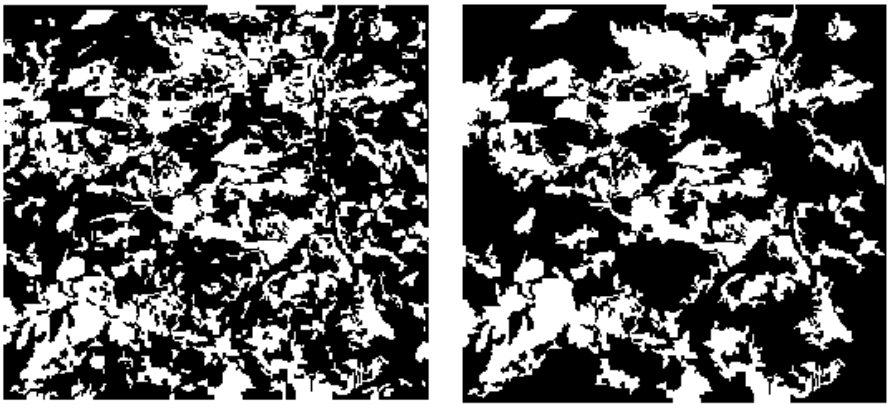
3.2 PIMTAS architecture

Input data is a cross country mobility map containing data on topography, vegetation, soil types, transportation, traversability, etc., plus information about the disposition of enemy and friendly forces and the location of weather fronts. For a given vehicle type, this input data is used to specify *go* and *no-go* regions on the map. For traversal across terrain, each map pixel has a cost penalty which is a linear combination of the traversal time which depends on slope, soil type, vegetation, soil moisture, etc. and a threat penalty which depends on its proximity to enemy threats (missiles, anti-aircraft guns, etc.) Map pixels that are too close to the top of a hill, a threat, or a weather front are labeled as points to be avoided by assigning them a high cost penalty.

Interactive Control: The Interactive Control module provides the military planner (e.g., a tank commander or helicopter pilot) a way to choose how much emphasis he wishes to place on each of the above factors. For example, if the planner wants to reach the goal quickly and is willing to use a dangerous route to do so, he would place a large weight on time and a small weight on threats. PIMTAS would then generate a direct path that may go close to enemy threats. On the other hand, if time is not a major factor and the planner's primary concern is safety, he would place a large weight on threats and a small weight on time, in which case PIMTAS would generate a path that stays as far away from threats as possible and consequently may be much longer.

Generating the Skeleton: The first step is to construct a *go/no-go* map from the original traversability map that was derived using the NATO Mobility Reference Model. Currently, areas are called *no-go* only if they are *water*, *obstacle* or *urban* areas. This last restriction follows military doctrine that urban areas are to be avoided. The grid-level planner will not be bound by the *no-go* representation at the graph level since it uses the multivalued traversability data. The final step is to remove small islands of *go* completely surrounded by *no-go* and also small islands of *no-go* completely surrounded by *go*. The user specifies whether eight-neighbor connectedness or four-neighbor connectedness is to be used to determine if a given blob is isolated or connected. The maximum size of a blob that can be removed is a program variable. The clean-up operation typically cuts the number of nodes in half. A node of the line-thinned skeleton may actually be located in a *no-go* island which was deleted when the binary map was constructed. In this case, the node is moved to the nearest *go* areas. The smoothed binary map is now ready to be thinned. Fig. 1(a) and (b) compare the original noisy binary map and the cleaned-up map. The map data is from an area near Lauterbach, Germany.

A newly developed single-pass line-thinner [23] was used to do the skeletonization. It does eight-neighbor connectedness along both diagonals and unlike other single-pass algorithms, it thins concave corners at the same rate at which it thins convex corners. Thus the right angle in the letter L is preserved when it is thinned. On an SGI 100/50 Indy workstation and using a GCC compiler, a 237 by 224 pixel image was thinned in less than two seconds.



(a) (b)

Figure 1: (a) is a binary traversability map and (b) is the cleaned binary traversability map.

Vectorization: The Raster-to-Graph module scans the skeleton to find all nodes and the connections between them. In some case, there are many arcs departing from a node and the node consists of two or more pixels. In an extreme case, a node may have seven arcs attached and with the node consisting of seven pixels.

Simplifying the Graph: Several procedures are used to simplify the graph. The ideal is to have a graph with a distance of at least 50 to 100 between pixels. The grid-level planner has the responsibility for doing planning over short distances. After the initial conversion to graph form, many leaf arcs are only a few pixels longs. These arcs are all pruned. Frequently, there are adjacent nodes which need to be merged into a single node. The node-merger module merges nodes if the separation of the nodes is less than a threshold value. As a result of the previous pruning and merging, some nodes will have only two arcs connected to them. Such nodes are eliminated unless there is a large change in direction between the two arcs incident to the node. A final simplification was to remove all leaf nodes.

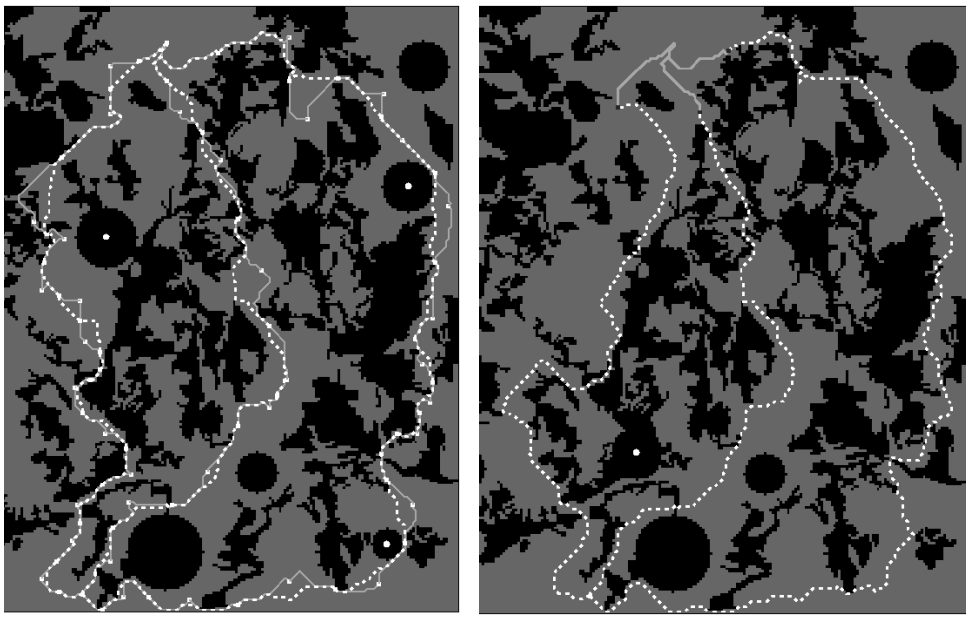
Grid-level Planner: The grid-level route planner computes paths between all nodes which were linked in the original skeleton. Since the paths that must be computed are typically short, several paths can be computed in less than a second. The planner uses the A* algorithm to compute the optimum path with the cost a weighted sum of the threat and traversal-time costs of each pixel traversed plus an *underestimation* of the cost to completion. The estimator function is the key quantity that determines how efficiently the algorithm works. The approach we used was to center a set of concentric annuli on the destination point with the spacing between annuli equal to that of the data grid. The cost to completion from some arbitrary point that is in the kth annulus from the center can not be less than the sum of the first k values of the vector $max\_value(n)$.

Graph-level Planner: The graph-level planner operates on the graph such as the one shown in Fig. 3 and uses the weights between nodes computed by the grid-level planner. The graph-level planner computes a user-specified number of independent paths. Such paths are defined to be paths that do not share any nodes or arcs. This requirement insures that two tank columns will not share the same path at the same time. The A* algorithm is used to compute the first path and as is normal with A*, it prunes the search tree of more costly paths to a given node. The complication comes with computing additional paths between the start and goal nodes. Arcs that form part of the first path are effectively removed from the graph when succeeding paths are computed. Support for pruning some node in the search tree may be a node which is part of the previously computed route. Since the node is not available for use in planning the current path, it can not provide support for pruning an element of the search tree. There are two steps in updating the linked list and search tree: (1) Remove from the linked list all nodes that have a father-ptr ancestor that is of type "route." The start node is marked non-terminal rather than route and thus does not cause the entire tree to be pruned. (2) The linked list is scanned for nodes that are marked pruned. For each pruned node, we traverse the down-ptr list until a node is located that has the same node-nbr as the pruned node. If this node is of type "route" then the pruned node is changed to "terminal." The final step is for the user to specify the starting position and the goal position.

Experimental Results: Fig. 2 is the end result of generating a binary map, thinning the map, converting it to a graph, simplifying the graph, using the grid-level planner to compute paths between all connected nodes in the graph and finally specifying a start point and an end point for computing three independent routes. We see that there are three independent routes computed from user selected start at the top of the figure and a destination point at the bottom. The gray regions of the map correspond to *go* areas and black represents *no-go* regions. It should be remembered that different gray areas can have large differences in allowable speeds and the optimum path may not be a straight line across a given gray area. The three unrelaxed routes are shown as solid white lines and the relaxed paths are indicated by dashed lines. The unrelaxed paths are constrained to go through the nodes of the original skeleton. The locations of some of these nodes can be seen in Fig. 2(a) by locating those points on the unrelaxed paths that differ most sharply from those of the relaxed paths. The three unrelaxed paths in Fig. 2(a) were on the average 18.5 percent more costly than the corresponding relaxed path. Fig. 2(b) shows paths with threats removed and added a fixed time after travel was initiated on the three routes.

4 New Approaches

This section discusses several new concepts which have not yet been implemented in PIMTAS. In the current implementation of PIMTAS, when multiple independent routes are requested, the system computes the best route between the starting



(a) (b)

Figure 2: (a)Three independent routes on combined mobility/threat map of area near Lauterbach, Germany. Threats can be identified by the black circular holes in the gray background. Small white circles within threat-circles identify threats removed in (b). (b) Same routes with one threat removed and three threats added. Small white circle within threat-circle indicate added threat. Start of dotted path indicates time at which threats changed and paths were recomputed.

point and the destination, removes the segments that make up the route from the linked list of nodes used for A* and computes the next best route. This procedure continues until the requested number of routes have been found or until the planner can no longer find a new route. The problem is that the first route is planned without regard to subsequent planning of routes. An algorithm is presented that guarantees that the sum of the cost of the specified number of routes is a minimum. In the next section the algorithm is developed from consideration of an eight node graph in which PIMTAS fails to compute an optimum solution. Using this graph, the requirements for a new algorithm that will compute an optimum set of routes is developed.

Sec. 4.2 describes HERMES (HEterogeneous Reasoning and Mediator System), which was developed at the University of Maryland. With HERMES combined with PIMTAS, we will be able to answer complex queries that require accessing a relational data base, a geographic information system as well as a route planner in order to answer a query. The salient features of HERMES are described and several example queries are presented.

4.1 Cooperative selection of multiple paths

The Hierarchical Route Planner, discussed in Sec. 3 has the ability to sequentially compute several non-competing optimum paths. Non-competing indicates that the selected routes do not share any arc-segments or nodes in the graph representation of mobility. It will occasionally be observed that if the first selected path were to be modified slightly, with only a small increase in traversal time, then the second path could use a significantly shorter path. In Fig. 3a, use of the A* algorithm to go from node 0 to node 7 results in the path 0-1-2-5-6-7 with a cost of 5. The second and third paths will be 0-4-7 and 0-3-7 each with a traversal cost of 10. The sum of the three traversals is 25. The corresponding search tree is shown in Fig. 3(b) for the case in which no heuristic knowledge is used. If we had instead chosen 0-1-4-7 as the first path with a greater cost of 7, then the other two paths would be 0-2-5-7 and 0-2-6-7 each with a cost of seven. The cost of the three paths is 21 compared to 25 previously. What we need is an algorithm that will simultaneously optimize all paths. Fig. 3(c) is a search tree which does give the optimum solution for the three paths. Comparing the search trees of Fig. 3(b) and Fig. 3(c), we see that the key difference is that in the first search tree, node 2 on the branch 0-1-2 caused pruning of node 2 on the branch 0-2. At the time this node was pruned, the A* algorithm had no way to know that the wrong pruning choice had been made in order to the optimize the sum of all the paths.

In order to simultaneously optimize two or more paths, we will need a separate search tree for each separate route. The maximum number of possible routes will be limited by the number of arcs connected to the start node and to the destination node. For the sake of simplicity, let whichever of these two nodes that has the fewest number of arcs attached be considered the start node and the other node the destination node. We will need to search each one of the arcs leaving the start node and generate the associated search tree. For convenience, all four search trees starting from node 0 of Fig. 3a are shown in Fig. 3(d) as if they were a single tree but with dotted lines from the start node to each of its connecting nodes. Without loss of generality, h(n) is assumed zero for all n. There can be only one independent route per search tree since each tree is characterized as having only one arc connected to the start node. The underlined nodes marked A.W. are precisely those nodes which would be pruned in an ordinary search tree. However, pruning leads to selection of the single-path-efficient route 0-1-2-5-6-7. The solution is to explore the nodes marked A.W. in an *alternative world*. Thus what would be pruning in a regular search tree is a collision here and all alternatives must be explored. Some terms must be defined. When a collision occurs between nodes from different search trees, one node is marked A.W. and let us call the other node the *collider* node. Let n_{req} be the number of requested routes and n_{arcs} be the number of arcs connected to the start node. Obviously n_{req} must be less than or equal to n_{arcs}. Using these definitions, some node $k_{A.W.}$ marked for *alternative world* exploration need never be explored if some world has already been explored in which n_{req} routes have been found and provided that the cost of each route is less than or equal to $f(k_{A.W.})$.

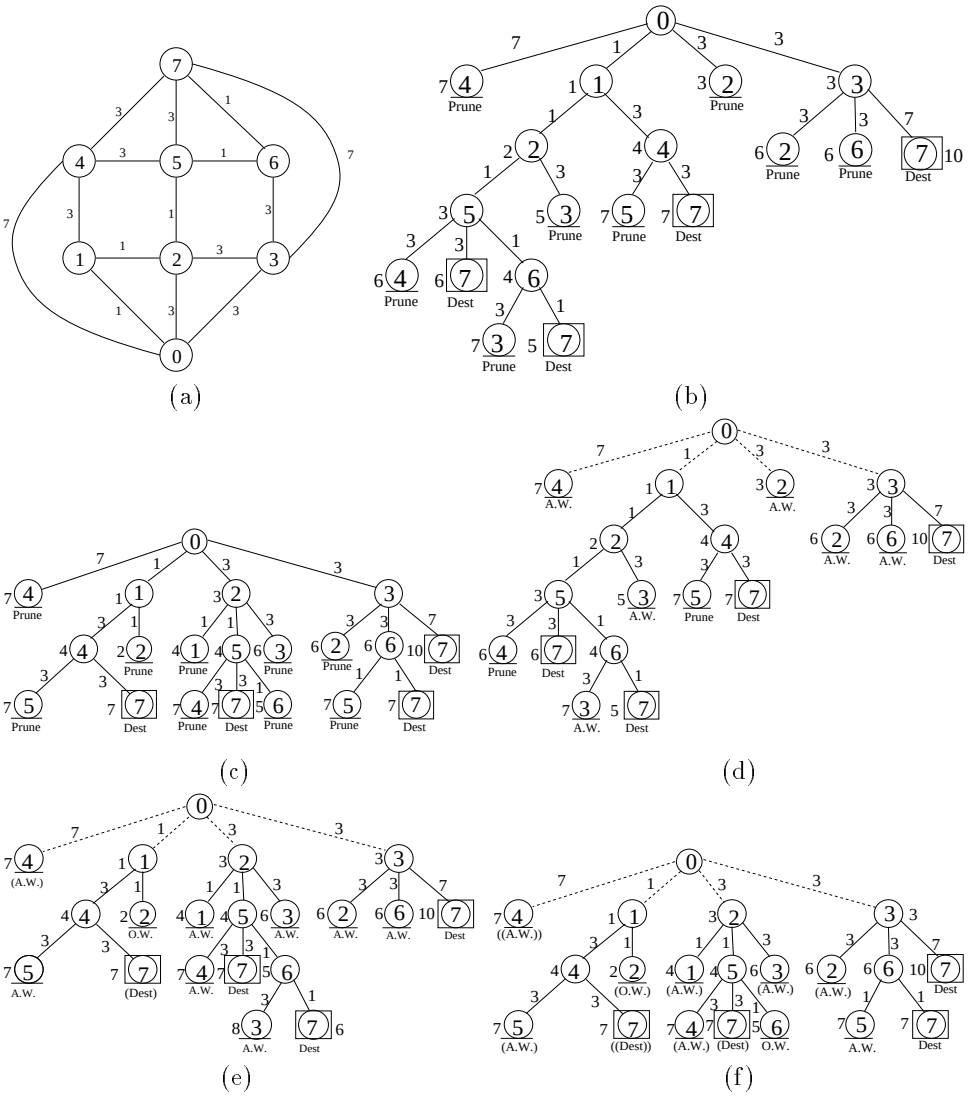


Figure 3: (a) Weighted graph that illustrates advantages of cooperative planning. Numbers on arc are weights of the arcs. (b) Search tree for graph of (a). (c) Desired search tree for cooperative multiple path selection. (d) Four search trees for descendents of Node 0 (start node) in (a). A.W. indicates node is awaiting *alternate world* expansion of node. (e) Search tree of *alternate world* from terminal node of path 0-2 of (d). O.W. indicates node which was expanded in the parent world but which was pruned in this alternate world. Parentheses around Dest, Prune, A.W. and O.W. indicate level of recursion at which the node was marked. One set indicates parent world and two sets indicate grandparent world. (f) Search tree of *alternate world* from terminal node of path 0-3-6 of (e).

Fig. 3(e) shows the result of the expanding the alternate world of the 0-2 node in Fig. 3(d). O.W. indicates node which was expanded in the parent world but which was pruned in this alternate world. Parentheses around Dest, Prune, A.W. and O.W. indicate level of recursion at which the node was marked. One set indicates parent world and two sets indicate grandparent world. When the 0-1-2 node was made a terminal node and its branches removed, the nodes 0-3-2 and 0-3-6 were no longer in collision with descendents of the 0-1-2 node. Now however, they are both in collision with the descendents of the 1-2 node in this *alternate world*. Note that the 0-4 node is now marked (A.W.) to indicate that the collision occurred in the parent world. Fig. 3(f) shows the result of expanding the *alternate world* of the 0-3-6 node in Fig. 3(e). In this world we have the three desired routes that optimally connect nodes 0 and 7.

We must now consider the rules which determine when we will explore an *alternate world* . If we wait too long, more backtracking may be required for the *collider* node in the *alternate world*. If we explore an *alternate world* node to soon, it may be the case that we needlessly explored that world. The answer is to treat the *alternate world* node as simply another node on the open list. Each time a world is discovered with the specified number of routes, the solution is returned and the cost of the routes is summed for comparison with other solutions found in other *alternate worlds*. For the set of solutions for a given world, let us define the cost of the highest cost solution to be f_{soln_max} The search is completed when the f cost of every node on the open list is greater than f_{soln_max} of some *alternate world*.

We still have one unanswered question: How do we do backtracking on the collider node? In the current implementation of the graph-level planner, the pointers of the search tree only permit traversal up the tree. We need to excise the nodes of the search tree that point to the collider node. A new pointer must be added, which will allow traversal down the tree. The excising of limbs can then be easily accomplished.

In summary, a separate search tree is maintained for each path, with normal pruning within a tree. When two search trees *collide*, the node which ordinarily would be pruned is marked A.W. and is saved for expansion in an alternate world. The A.W. node will also have a pointer to the collider node. The A.W. node is added to the open list and the *alternate world* is explored when it is popped from the open list. When an *alternate world* is created, the limbs of the collider node are traced and removed from the search tree. Any other nodes which were marked as A.W. or pruned because of one of the excised nodes will become a terminal node in the *alternate world*. Operation of the A* now continues on the modified search tree. and the conditions for termination of the search remain unchanged from the algorithm described in Sec. 3.2. When the search of an alternate world is completed, and if the number of independent routes is greater than m_{req} then f_{soln_max} is computed for the set of solutions found in that world. When unwinding the recursion, if all nodes on the open list of the parent world are greater than the f_{soln_max} from the descendent world, than search in the parent world is terminated and the unwinding

continues until the top-level world is reached. When the search terminates in that world, the overall search is completed and the best solution set has been found.

4.2 Intelligent Terrain Reasoning in HERMES

HERMES (HEterogeneous Reasoning and Mediator System) [24] is a system that has been developed at the University of Maryland to facilitate the development and rapid deployment of mediators for different kinds of applications. It uses the Hybrid Knowledge Base paradigm, due to Lu, Nerode and Subrahmanian [7] to provide deductive database support for multiple modes of reasoning and multiple types of data.

HERMES also incorporates a *mediator development toolkit* that allows the *mediator author* to rapidly develop interfaces between HERMES and disparate data sources (called domains in HERMES). The language in which the mediator is written is a simple rule-based language with certain specific constructs, and with a special compiler that can be used to implement these special constructs. A rule is a statement of the form

$$\begin{aligned} A \quad \leftarrow \quad & \Xi_1 \,\&\, \ldots \,\&\, \Xi_m \,\| \\ & B_1 \,\&\, \ldots \,\&\, B_n \end{aligned}$$

where each of $A, B_1, \ldots, B_n$ are atoms, in the sense of logic and each of $\Xi_1, \ldots, \Xi_m$ is a special atom of the form

$$\texttt{in}(\texttt{X}, \texttt{domainname} : \langle \texttt{domainfunction} \rangle (\langle \texttt{arg1}, \ldots \texttt{argk} \rangle)).$$

The predicate *"in"* is used to query external data bases or programs such as PIMTAS. The mediator may, for instance, contain a call to PIMTAS of the form

$$\texttt{in}(\texttt{Route}, \texttt{pimtas} : \texttt{route}((35,70),(200,98)))$$

This atom succeeds just in case the variable symbol, `Route` is instantiated to one of the routes between $(35, 70)$ and $(200, 98)$ that is returned by PIMTAS.

We can now see how we could answer a complex query in which we want to find a route to a currently unknown destination. We need to get to a place that has an airfield as well as certain types of ammunition. Presumably these resources are needed in order for the autonomous/manned vehicle to satisfy its mission. Note, in particular, that this may *not* have been the initial mission of the vehicle – it may be the case that the battlefield situation has changed drastically since the original mission plan was constructed, and this reflects a new decision taken by the personnel (if any) operating the vehicle. The salient feature about this example is that the *identity of the destination is unspecified, though the properties that a suitable destination should satisfy are specified.* In order for an appropriate solution to be found to this problem, it may be necessary to access heterogeneous databases distributed at different sites.

Solving complex queries using HERMES and PIMTAS

Let us now see how the complex examples described earlier may be solved within the HERMES framework, using PIMTAS as a domain. For this, let us suppose that we have one relational database(e.g. Paradox) containing a relation called `facilities` having the schema (Name,Facility). Thus, this relation may contain a tuple of the form $(awasa, airport)$ denoting that the place, Awasa, has an airport. Other tuples in the relation `facilities` may be similarly interpreted. There may be another database (e.g. DBASE) containing a relation called `supplies` having the schema (Place,Item) – an example tuple in this relation is $(Awasa, gas)$ specifying that gas is available at Awasa[b].

HERMES may now be used to integrate *three domains* – PIMTAS, DBASE, and PARADOX. In addition, a fourth `spatial` domain that identifies points (xy-coordinates with place names) is needed to solve the queries posed above.

Query: Let `rte1` be a ternary predicate such that `rte1`(O, D, R) is satisfied iff R is a route from the origin to an unspecified destination such that the destination has an airfield as well as certain types of ammunition. For this, we may define the following clause in the mediator:
`rte1(O,D,R)` $\leftarrow$

$$\begin{aligned}
&\texttt{in(P1, paradox : select}_{=}\texttt{(facilities, facility, ``airfielddd''))} \,\&\\
&\texttt{in(P2, dbase : select}_{=}\texttt{(supp, item, ``ammunition'')} \,\&\\
&= \texttt{(P1.place, P2.place)} \,\&\\
&\texttt{in(D, spatial : findpt(P1.place))} \,\&\\
&\texttt{in(R, pimtas : route(O, D))}.
\end{aligned}$$

Suppose the vehicle is at location ℓ; then it can ask the query

$$\leftarrow \texttt{rte1}(\ell, \texttt{D}, \texttt{R}).$$

This is then processed as follows: `PARADOX` is invoked and asked to `SELECT` all tuples from the `facilities` relation that have the `facility` field set to `airfielddd` Subsequently, `P1` gets instantiated to one of these selected tuples. `DBASE` is then asked to `SELECT` all tuples form the supplies or `supp` relation that have the `item` field set to `ammunition`. `P2` is instantiated (or points to) one such tuple. We then check if `P1` and `P2` have the same `place` field – that is, have we found a *single* place with ammunition and having an airfield? If not, the HERMES inference engine looks for other possible instantiations of `P1` and `P2` that satisfy these constraints. After such instantiations are obtained, the xy-location of the place `P1.place` (which is the same as `P2.place` is computed using the spatial domain. . `D` gets instantiated to this xy-location, i.e. D is a pair (x, y), and PIMTAS is then called to find a route

[b] In reality, these relations are somewhat more detailed, but we keep them simple here in order to facilitate an easy presentation.

from the origin (i.e. the place where the autonomous/manned vehicle is currently located) to the place D. □

We have now shown how the HERMES system may be used, in conjunction with systems such as PIMTAS, to solve problems that neither could solve in isolation. The reader who is interested in a detailed account of HERMES is referred to the following papers [24, 6, 7, 8].

5 Conclusions

In Sec. 2, we reviewed the advantages and limitations of the various methods of decomposing the map plane in order to efficiently search the map plane. Although a number of these methods have elements of a hierarchical organization, none of them provide the capability to efficiently plan high resolution routes over long distances. In Sec. 3, we described a hierarchical implementation that can efficiently search a large map plane without loss of fine resolution detail.

Intelligence is being added to the route planner by linking the planner to an asynchronous production system[21] built on top of the CLIPS [22] expert system shell developed by NASA. The expert system will provide intelligent and adaptive control for rapid, real time responses to unexpected, real time events. It can, for example, analyze the obstacles and determines places where use of obstacle breeching equipment could significantly reduce the length of an optimum route. The HERMES system described in Sec. 4.2 provides a capability to easily interface the route planning system to diverse databases and Geographic Information Systems (GIS) and to ask complex queries that require interrogation of the GIS and database as well as to a route planner in order to obtain an answer. This capability was recently demonstrated.

Acknowledgements

We thank Anne Brink for writing the initial grid-level route planner which was the basis for the design of the PIMTAS grid-level planner and we thank William Seemuller for writing software used to remove noise from the binary *Go/NO-GO* map. This work was supported by the Army Research Office under Grant Nr. DAAL-03-92-G-0225 and by the National Science Foundation under Grant Nr. IRI-9109755. Partial funding for this work was also provided by the U. S. Army Corps of Engineers.

References

[1] Richard B. Ahlvin and Peter W. Haley, *NATO reference mobility model, edition II users guide*, Waterways Experiment Station, Vicksburg, MS, (1992).

[2] Howard A. Zebker, Charles Wermer, Paul Rosen and Scot Hensley, *Accuracy of topographic maps derived from ERS-1 interferometric radar*, IEEE Trans. on Geoscience and Remote Sensing, **32** (1994) 823–836.

[3] John Benton, *Hierarchical route planner*, Proc. Applications of Artificial Intelligence VI, SPIE, **937**, (Apr. 1988) 428–433.

[4] John Benton and Anne Brink, *Hierarchical route planner*, Proceedings of the 1990 Army Science Conference, Vol. 1, 87–97.

[5] G. Wiederhold. *Intelligent integration of information,* Proc. 1993 ACM SIGMOD Conf. on Management of Data, 434–437.

[6] S. Adalı and V.S. Subrahmanian. *Amalgamating knowledge bases, II: algorithms, data structures and query processing*, Univ. of Maryland, **CS-TR-3124** (1993). Accepted for publication in: Intl. J. of Intell. Coop. Info. Sys.

[7] J. Lu, A. Nerode, and V.S. Subrahmanian, *Hybrid knowledge bases*, Univ. of Maryland, **CS-TR-3037**. Accepted for publication in IEEE Trans. on Knowledge and Data Engrng.

[8] V.S. Subrahmanian, *Amalgamating knowledge bases*, ACM Trans. on Database Sys. **19**, 1994 291–331.

[9] Joseph Mitchell, *An algorithmic approach to some problems in terrain navigation*, AI, **37** (1988) 171–197.

[10] O. Khatib, *Real-time obstacle avoidance for manipulators and mobile robots*, Int'l J. of Robotics Res. **5** (1986) 90–98.

[11] Jean-Claude Latombe, *Robot motion planning*, Kluwer, 1991.

[12] Nils Nilsson *A mobile automaton: an application of artificial intelligence techniques* Proc. IJCAI **1** Washington, DC (1969) 509–520.

[13] Michael Longtin and Dalila Megherbi, *Concealed routes in ModSAF*, Proceedings of the 5th Conf. of Comp. Gen. Forces and Behav. Repr., Orlando, FL, (May 1995) 305–313.

[14] H. Blum, *A transformation for extracting new descriptors of shape*, Models for the Perception of Speech and Visual Form, MIT Press, Cambridge, (1967) 362–380.

[15] D. R. Powell, J. C. Wright, G. Slentz, and P. Knudsen, *Representations to support reasoning on terrain*, U. S. Army Symposium on Artificial Intelligence Research for Exploitation of the Battlefield Environment, El Paso, Texas, (1988), 212–222.

[16] Jed Marti, *Cooperative autonomous behavior of aggregate units over large scale terrain*, Proc. Simulation and Planning in High Autonomy Systems, IEEE Comp. Soc. Press, (Mar. 1990) 58–64.

[17] Joseph Mitchell, David Peyton and David Kiersey, *Planning and reasoning for autonomous vehicle control*, Int'l J. of Intell. Sys. **2** (1987) 129–189.

[18] Tom Kreitzberg, Ted Barragy, and Nevin Bryant, *Tactical movement analyzer: a battlefield mobility tool*, Proceedings, Fourth Joint Tactical Fusion Symp., Laurel, MD, (1990).

[19] P. N. Stiles and I. S. Glickstein, *The evolution of PRP search algorithm*, IBM J. of R. and D. **38** (1994) 167–181.

[20] Robert Richbourg, *Solving a class of spatial reasoning problems: minimal-cost path planning in the cartesian plane*, Ph. D. Thesis, Naval Postgraduate School (1987).

[21] Arvind Sabharwal, S. Sitharama Iyengar, C. R. Weisbin and F. G. Pin, *Asynchronous production systems*, J. of Knowledge Based Sys. (1989) 122–132.

[22] Joseph Giaratano, *Expert Systems: Principles and Programming*, PWS Kent, Boston (1989).

[23] Weian Deng, Sitharama Iyengar and Nathan Brener, *A fast parallel thinning algorithm for the binary image skeletonization*, Submitted for publication.

[24] S. Adalı R. Emery, J. Lu, A. Rajput, T.J. Rogers, R. Ross, and V.S. Subrahmanian, *HERMES: A heterogeneous reasoning and mediator system*, draft manuscript.