J. Indian Inst. Sci., May–Aug. 2004, **84**, 67–76
© Indian Institute of Science.

Three-dimensional route planning for large grids

NATHAN E. BRENER[1], S. SITHARAMA IYENGAR[1]*, HUA C. LOONEY[1],
NARAYANADAS VAKAMUDI[1], DECAI YU[1], QIANYU HUANG[1] AND JACOB BARHEN[2]
[1]Robotics Research Laboratory, Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803, USA. *Visiting Satish Dhawan Chaired Professor, Indian Institute of Science, Bangalore 560 012.
[2]CESAR Laboratory, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA.
*Email: iyengar@bit.csc.lsu.edu; Phones: +1-225-578-1252; Fax: +1-225-578-1465.

Received on July 23, 2003; Revised on August 16, 2004.

Abstract

We have developed a 3D route planner, called 3DPLAN, which employs the A* algorithm to find optimum paths. The A* algorithm has a major advantage compared to other search methods because it is guaranteed to give the optimum path. In spite of this significant advantage, no one has previously used A* in 3D searches as far as we are aware. The probable reason for this is the belief that the computational cost of using A* for 3D route planning would be prohibitive. In this paper we show that, on the contrary, it is quite feasible to use A* for 3D searches if one employs the new mobility and threat heuristics that we have developed. These new heuristics substantially speed up the A* algorithm so that the run times are quite reasonable for the large grids that are typical of 3D searches.

Keywords: 3D route planner, A* algorithm, 3D searches, mobility and threat heuristics.

1. Introduction

Route planning has been widely used for civilian and military purposes. Three-dimensional (3D) route planning is especially useful for the navigation of autonomous underwater vehicles (AUVs) and combat helicopters. 3D route planning is a very challenging problem because the large grids that are typically required can cause a prohibitive computational burden if one does not use an efficient search algorithm. Several search algorithms have been proposed to perform 3D route planning, including case-based reasoning and genetic algorithms.

Case-based reasoning [1, 2] relies on specific instances of past experience to solve new problems. A new path is obtained by searching previous routes to see if there is one that matches the current situation in the features, goals and constraints. The new path is generated by modifying an old path in the previous path database using a set of repair rules. However, since the number of possible threat distributions is very large for most battle areas, it would not be feasible to store old routes for all or most of the possible threat arrangements. Thus the case-based method is not suitable for including threats in route planning. In addition, when it has to synthesize complete new routes (in an area where no old paths are available) or modify old routes by synthesizing new segments, it doesn't use a

*Author for correspondence.

guaranteed best-first search algorithm such as A*, but instead uses straight line segments that go around obstacles. Thus the routes that it generates are neither locally nor globally optimal.

The genetic algorithm (GA) method [3–6] is a stochastic search technique based on the principles of biological evolution, natural selection and genetic recombination. Genetic algorithms generate a population of solutions. Then such solutions mate and bear offspring solutions in the next generation. In this way, the solutions in the population improve over many generations until the best solution is obtained. However, genetic algorithms have been accepted slowly for research problems because crossing two feasible solutions does not, in many cases, result in a feasible solution as an offspring. The other disadvantage of GA is that although it can generate solutions to a route planning problem, it cannot guarantee that the solution is optimal (i.e. it can converge to a local, rather than a global, minimum).

The A* algorithm [7] is a guaranteed best-first search algorithm that has been used previously in 2D route planning by several researchers including some of the authors of this paper [8, 9]. A major advantage of the A* algorithm compared to the other methods mentioned above is that A* is guaranteed to give the optimum path. In spite of this significant advantage, no one has previously used the A* algorithm for 3D route planning as far as we are aware. The probable reason for this is the belief that the computational cost of using A* for 3D route planning would be prohibitive. In this paper we show that, on the contrary, it is quite feasible to use A* for 3D searches if one employs the new mobility and threat heuristics that we have developed. These new heuristics substantially speed up the A* algorithm so that the run times are quite reasonable for the large grids that are typical of 3D searches.

2. A* Algorithm and our implementation

2.1. *Mobility and threat maps*

Brener and Iyengar, in collaboration with John Benton, have previously developed a 2D A* route planner [8, 9], called the Predictive Intelligence Military Tactical Analysis System (PIMTAS), for military terrain vehicles such as tanks. Figure 1 shows an example of a 2D mobility map (top) and threat map (bottom) that were used as input to PIMTAS. The upper map in the figure is an actual mobility map of an area near Lauterbach, Germany, and the lower one is a prototype threat map which was generated in order to test the program. Both maps have 237×224 grid points. The mobility map has four types of GO regions represented by the colors green, light green, yellow-orange, and orange, which denote unlimited, limited, slow, and very slow areas, respectively. The mobility penalty for grid points in the unlimited, limited, slow, and very slow regions is 1, 2, 3, and 4, respectively. Thus the minimum mobility penalty at each grid point is 1. In general, the mobility map has three types of NO-GO regions represented by the colors red, blue, and white, which denote impassable obstacles, water, and urban areas, respectively. This last restriction follows military doctrine that urban areas are to be avoided. In the prototype threat map, each threat is modeled by a red inner circle where the vehicle is not allowed to go since it would almost certainly be destroyed if it came that close to the threat, and an orange outer circle where

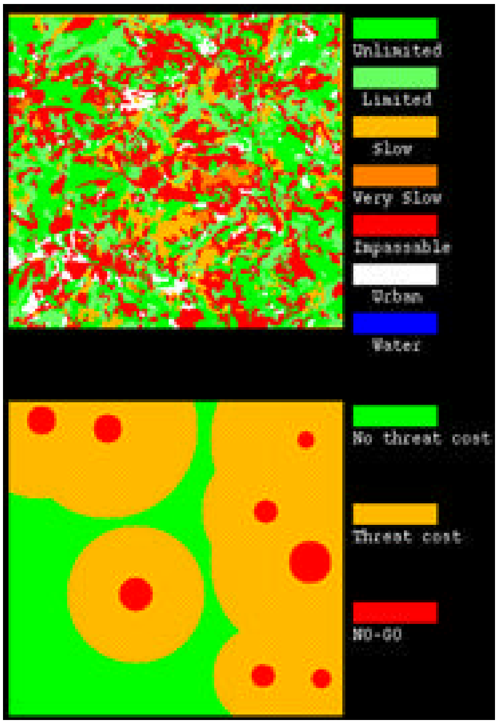


FIG. 1. 2D mobility map (top) and threat map (bottom) used as input to PIMTAS.

the vehicle is within the range of the threat. The green regions are outside of the range of all of the threats. The threat penalty for each threat varies linearly from 1 to 0 as one goes radially outward from the boundary of the red circle to the boundary of the orange circle. If a grid point is within the orange circle of more than one threat, the threat penalty at that point is the sum of the threat penalties of all of the threats acting on that point. Grid points in the green regions have a threat penalty of 0.

In our new 3D route planner, which will be called 3DPLAN in this paper, the 2D mobility and threat maps described above have been extended to 3D in order to test the program. In 3DPLAN, the search region is represented by a digital map consisting of a Cartesian grid of points in which the step size in the x and y directions is the same but the step size in the z direction is in general different. Both the 3D mobility map and the 3D threat map have $237 \times 224 \times 150$ grid points in the x, y, and z directions for a total of almost 8 million points. To our knowledge, this is the largest number of grid points that has ever been used in an A* search. In the 3D mobility map, each grid point in the GO regions has a mobility penalty of 1, 2, 3, or 4 depending on the mobility conditions. Grid points that are located in impassible areas are labeled as avoided points where the vehicle is not allowed to go. This test mobility map will be replaced with a more realistic map for underwater vehicles once we obtain actual oceanographic data. In our prototype 3D threat map, each threat is modeled by an inner sphere where the vehicle is not allowed to go and an outer sphere where the vehicle is within the range of the threat. For each threat, the threat penalty varies linearly from 1 to 0 as one goes radially outward from the surface of the inner sphere to the surface of the outer sphere. As in the 2D case, if a grid point is within the range of more than one threat, its threat penalty is the sum of the penalties of all of the threats acting on that point. Grid points that are outside the range of all of the threats have a threat penalty of 0.

Given these maps, a starting point, and a target to be reached, the military planner enters a weight for each of the two path cost factors being considered: (i) mobility and (ii) threats. 3DPLAN will then quickly generate the lowest cost path from the starting point to the target, where the cost of the path is determined by multiplying the weight for each factor by the accumulated penalty for that factor. By entering a particular set of weights, the military planner can put any desired degree of emphasis on each of the cost factors. For example, a large weight for mobility and a small weight for threats would produce a fast path that may go close to enemy threats, while a large weight for threats and a small weight for mobility would produce a path that stays as far away from threats as possible and consequently may require a considerably longer travel time.

2.2. *A* Algorithm*

3DPLAN employs the A* algorithm, in which the total cost, f, of the path that goes through a particular grid point on the digital map (this grid point will be referred to as the current point) is given by

$$f = g + h \tag{1}$$

where g is the actual cost that was accumulated in going from the starting point to the current point and h is an *underestimate* of the remaining cost required to go from the current point to the target. The heuristic h is the key quantity that determines how efficiently the algorithm works. h must not only be a guaranteed underestimate of the remaining cost, which ensures that no potential optimum paths will be discarded due to overestimating their total cost, but must also provide as close an estimate as possible of the remaining cost. The closer h is to the actual remaining cost, the faster the algorithm will find the optimum path. Thus the success of the algorithm depends on the choice of the heuristic h. With a proper choice of h, the algorithm can be highly efficient and can find the optimum path in a matter of seconds.

The actual accumulated cost, g, is given by

$$g = a_m M + a_t T \tag{2}$$

where M is the accumulated mobility penalty, T is the accumulated threat penalty, a_m is the mobility weight, and a_t is the threat weight. The weights a_m and a_t are entered by the military planner, as discussed above.

The accumulated mobility and threat penalties are given by

$$M = \Sigma R_i(M_{i-1} + M_i)/2 \tag{3}$$

$$T = \Sigma R_i(T_{i-1} + T_i)/2 \tag{4}$$

where the sum is over the grid points traversed in going from the starting point to the current point, M_i is the mobility penalty at grid point i, T_i is the threat penalty at grid point i, and R_i is the stepsize to go from grid point i–1 to grid point i.

The heuristic, which is an underestimate of the remaining cost required to go from the current point to the target, is given by:

$$h = a_m h_m + a_t h_t \tag{5}$$

where h_m, the mobility heuristic, is an underestimate of the remaining mobility penalty and h_t, the threat heuristic, is an underestimate of the remaining threat penalty. We now describe the mobility heuristic and threat heuristic in detail.

2.3. *Mobility heuristic*

Our mobility heuristic is different and better than the straight line heuristic used in previous A* approaches, since it is larger than the straight line heuristic and still is an underestimate of the remaining cost. We will first describe our mobility heuristic in 2D and then extend it to 3D.

Figure 2 shows a 2D mobility map that consists of a square grid of points. The step size in the x and y directions will be denoted by $r1$ and the diagonal step size will be denoted by $r2$, where $r2$ = sqrt(2)*$r1$. In this figure, each grid point has a mobility penalty of either 1, 2, 3, or 4 (i.e. 1 is the minimum mobility penalty at each grid point), and P1, P2 and P3 are the start point, current point, and target point, respectively.

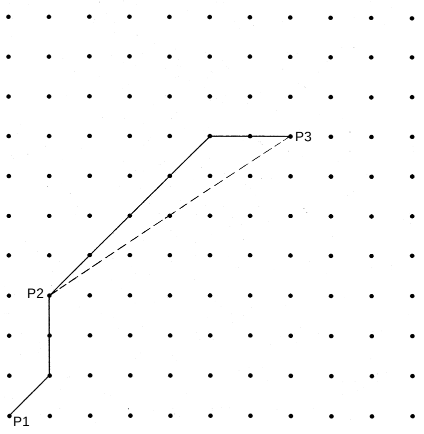
Let n_x and n_y be the number of steps in the x and y directions, respectively, between the current point and the target. Then the mobility heuristic h_m is given by:

$$h_m = \begin{cases} n_y r2 + (n_x - n_y) r1 & \text{for } n_x > n_y \\ n_x r2 + (n_y - n_x) r1 & \text{for } n_y \geq n_x \end{cases} \tag{6}$$

In the example given in Fig. 2, the mobility heuristic to go from the current point P2 to the target P3 is $h_m = 4*r2 + 2*r1$. This is larger than the straight line distance from P2 to P3, which is what other authors have used as the heuristic, and is still a guaranteed underestimate of the remaining mobility penalty to go from P2 to P3. Thus our mobility heuristic will cause the A* algorithm to run faster than it would with a straight line heuristic.

It is straightforward to extend this mobility heuristic to 3D. Figure 3 shows a cell in the 3D map in which the step size in the x and y directions is the same but the step size in the z direction is in general different. The figure shows the five possible step sizes, labeled $r1$, $r2$, $r3$, $r4$ and $r5$, that the vehicle can take to go from a grid point to one of its neighbors, where

$r1$ = step size in x and y directions
$r2$ = sqrt(2)*$r1$
$r3$ = step size in z direction
$r4$ = sqrt($r1$^2 + $r3$^2)
$r5$ = sqrt($r2$^2 + $r3$^2)



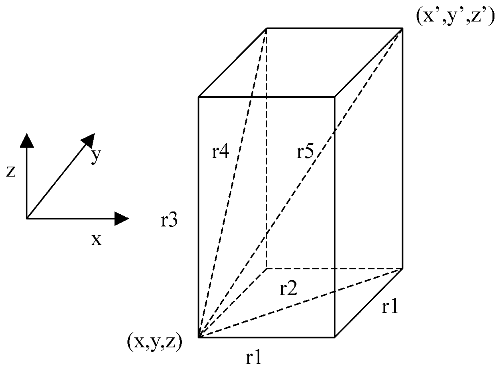


FIG. 2. Mobility heuristic from P2 to P3 (solid line). FIG. 3. Step sizes in the 3D map.

Let n_x, n_y and n_z be the number of steps in the x, y and z directions, respectively, between the current point and the target. The 3D mobility heuristic h_m is then given by

$$h_m = \begin{cases} n_x*r5 + (n_y - n_x)*r4 + (n_z - n_y)*r3 & \text{for } n_x \le n_y \le n_z \\ n_x*r5 + (n_z - n_x)*r4 + (n_y - n_z)*r1 & \text{for } n_x \le n_z \le n_y \\ n_y*r5 + (n_x - n_y)*r4 + (n_z - n_x)*r3 & \text{for } n_y \le n_x \le n_z \\ n_y*r5 + (n_z - n_y)*r4 + (n_x - n_z)*r1 & \text{for } n_y \le n_z \le n_x \\ n_z*r5 + (n_x - n_z)*r2 + (n_y - n_x)*r1 & \text{for } n_z \le n_x \le n_y \\ n_z*r5 + (n_y - n_z)*r2 + (n_x - n_y)*r1 & \text{for } n_z \le n_y \le n_x \end{cases} \tag{7}$$

2.4. *Threat heuristic*

As far as we know, a threat heuristic has never been used in the A* algorithm. The probable reason for this is that the minimum threat penalty is 0 rather than 1 (the minimum mobility penalty is 1 in our test map). Thus if one tried to use the same technique for the threat heuristic as was used for the mobility heuristic, the threat heuristic would be 0. In this paper, we present a threat heuristic that is different from the mobility heuristic and is, in general, nonzero. Thus our new threat heuristic will speed up the A* algorithm compared to not having a threat heuristic at all.

We will first describe our threat heuristic in 2D and then extend it to 3D. Figure 4 shows the same square grid of points that was shown in Fig. 2, where P1, P2 and P3 are the start, current and target points, respectively. The threat heuristic h_t will be an underestimate of the remaining threat penalty to go from the current point to the target point. In order to construct this threat heuristic, we draw concentric squares around the target; these squares are labeled 1, 2, 3…, as shown in the figure, and the target point is square 0. In order to go from the current point to the target, the vehicle must visit each square at least once (i.e. it must visit at least one point in each square). In order to ensure that the threat heuristic is a guaranteed underestimate, we will use the minimum threat penalty in the square as the threat penalty of the point that the vehicle visits. The threat heuristic is then given by

$$h_t = r1(T_c + T_{n,min})/2 + \Sigma r1(T_{i,min} + T_{i-1,\,min})/2 \qquad (8)$$

where T_c is the threat penalty of the current point, $T_{i,min}$ is the minimum threat penalty in square i, n is the number of squares between the current point and the target, the sum is over all squares between the current point and the target, and we have multiplied by the smaller of the two step sizes that the vehicle can take, $r1$, in order to ensure that the threat heuristic is an underestimate.

It is straightforward to extend this 2D threat heuristic to 3D. In 3D, we will use concentric rectangular boxes around the target. The vehicle will then have to visit each box at least once to go from the current point to the target. For the large boxes, the minimum threat penalty in the box may often be 0, depending on the distribution of threats. However, for small boxes (the ones close to the target), the minimum threat penalty in the box is less likely to be 0, especially if there is a dense distribution of threats around the target. In these cases, the threat heuristic will significantly speed up the A* algorithm.

2.5. *Dynamic threats*

3DPLAN can also handle dynamic (changing) threats. If new threats appear or known threats disappear or move while the vehicle is traveling along its path, the military planner can quickly update the threat map. 3DPLAN will then rapidly generate a new optimum path from the current position to the target.

3. Test calculations and comparison of heuristics

In this section we give the results of some optimum path calculations using our new 3D A* route planner, 3DPLAN. In these sample calculations we used two different mobility maps, labeled M1 and M2, and two different threat maps, labeled T1 and T2. In the mobility map M1, all of the grid points in the GO areas have a mobility penalty of 1 (i.e. the mobility is uniform except for the obstacles), while in mobility map M2, the points in the GO areas have a mobility penalty of either 1, 2, 3, or 4. The threat maps T1 and T2 contain 24 threats and 26 threats, respectively. All four of these maps have $237 \times 224 \times 150$ grid points for a total of almost 8 million points, which is the largest number of grid points that has ever been used in an A* search as far as we are aware. We used four different combinations of these maps: M1T1, M1T2, M2T1, and M2T2. For each of these combinations of a mobility map and a threat map, we calculated three different paths by choosing three different pairs of start/target points (Table I). Thus, altogether we calculated 12 different optimum paths in

Table I
Paths for test calculations

Path	Start point	Target
1	(6, 22, 0)	(236, 218, 149)
2	(6, 22, 0)	(136, 218, 89)
3	(100, 0, 50)	(236, 218, 149)

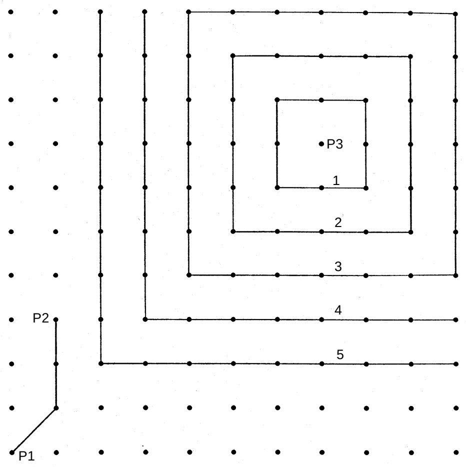


FIG. 4. Concentric squares in a 2D threat map.

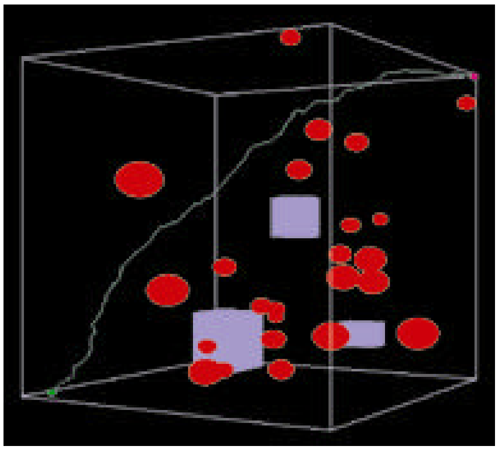


FIG. 5. Path 1 for the map combination T1M2.

a search space of approximately 8 million grid points in order to test our new program. In all of the path calculations, the mobility weight a_m and the threat weight a_t were both set equal to 1. For each of the 12 optimum paths, we did four different calculations using the following four combinations of the mobility heuristic and threat heuristic:

(1) Our new mobility heuristic, our new threat heuristic
(2) Our new mobility heuristic, no threat heuristic
(3) Straight line mobility heuristic, our new threat heuristic
(4) Straight line mobility heuristic, no threat heuristic

Combinations 1, 2 and 3 enable us to compare our new mobility and threat heuristics with the heuristic that other authors have used in A* searches (combination 4).

These optimum path calculations were done on a Dell PC with a 4.3 GHZ Pentium IV processor. One of these optimum paths, Path 1 for the map combination T1M2, is shown in Fig. 5. The spheres in this figure are the inner spheres surrounding the threats, the rectangular solids are the obstacles in the mobility map, and the optimum path goes from the lower left to the upper right.

Table II gives the CPU time in seconds for the optimum path calculations described above. It shows that when our new mobility and threat heuristics are used, all of the path calculations require less than a minute of CPU time. It also shows that for the 12 test paths considered, our new heuristics cause the route planner to run up to 4.3 times faster, compared to a straight line mobility heuristic and no threat heuristic, which is what other researchers have used in A* searches. In addition, it shows that our new mobility heuristic alone causes the planner to run up to 3.2 times faster and our new threat heuristic alone causes the planner to run up to 1.2 times faster, compared to a straight line mobility heuristic and no threat heuristic, respectively. These results demonstrate that our new mobility and threat heuristics significantly speed up the A* algorithm.

Table II
CPU time in seconds for optimum path calculations

	Map T1M1			Map T1M2			Map T2M1			Map T2M2		
	Path 1	Path 2	Path 3	Path 1	Path 2	Path 3	Path 1	Path 2	Path 3	Path 1	Path 2	Path 3
Our mobility heuristic, our threat heuristic	18.6	7.8	9.8	54.7	9.8	17.3	6.8	26.1	13.4	49.6	33.2	33.6
Our mobility heuristic, no threat heuristic	23.6	11.0	12.9	59.5	11.9	20.8	9.1	30.8	16.0	52.3	36.8	36.3
Straight line mobility heuristic, our threat heuristic	44.9	19.6	22.7	71.2	19.2	30.8	24.9	35.8	29.6	66.4	41.0	48.8
Straight line mobility heuristic, no threat heuristic	51.3	22.6	26.4	75.4	21.9	35.1	28.9	41.0	33.1	69.7	45.8	51.8

4. Conclusion

We have developed a 3D A* route planner, called 3DPLAN, which runs efficiently for the large grids that are typical of 3D maps. The A* algorithm has a major advantage compared to other search methods because it is guaranteed to give the optimum path. To our knowledge, this is the first time that A* has been used in 3D searches. The probable reason for this is that most researchers think that the computational cost of using A* for 3D route planning would be prohibitive. We have shown that, on the contrary, it is quite feasible to use A* for 3D searches as a result of the new mobility and threat heuristics that we have developed. These new heuristics substantially speed up the A* algorithm and make it a useful and efficient method for 3D route planning.

In the future we plan to test 3DPLAN with real oceanographic and battlefield data and adapt it to autonomous underwater vehicles and combat helicopters.

Acknowledgments

This work was supported in part by DOE-ORNL grant no. 4000008407 and by an NSF grant.

References

1. C. Vasudevan and K. Ganesan, Case-based path planning for autonomous underwater vehicles, *Autonomous Robots*, **3**, 79–89 (1996).
2. M. Kruusmaa and B. Svensson, Combined map-based and case-based path planning for mobile robot navigation, *Proc. Int. Symp. on Intelligent Robotic Systems,* January 10–12, 1998.
3. J. Smith and K. Sugihara, GA toolkit on the web, *Proc. 1st Online Workshop on Soft Computing*, August 1996, pp. 93–98.
4. K. Sugihara and J. Yuh, GA-based motion planning for underwater robotic vehicles, *Proc. 10th Int. Symp. on Unmanned Untethered Submersible Technology,* Autonomous Undersea Systems Institute, Durham, NH, 1997, pp. 406–415.
5. D. Kang, H. Hashimoto and F. Harashima, Path generation for mobile robot using genetic algorithm, *Trans. Inst. Electl Engrs Jap.,* **117-C**, 102–109 (1997).
6. F. Jensen and U. Kiarulff, DAT5 project proposal–self-adaptive genetic algorithm, Research Unit of Decision Support Systems, Aalborg University, Denmark, www.cs.auc.dk/research/DDS/dat5_projekter/genalg.html
7. P. Hart, N. Nilsson and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Trans. Systems Sci. Cybernetics,* **SSC-4**, 100–107 (1968).
8. J. R. Benton, S. S. Iyengar, W. Deng, N. E. Brener, and V. S. Subrahmanian, Tactical route planning: New algorithms for decomposing the map, *Int. J. Artif. Intell. Tools*, **5**, 199–218 (1996).
9. N. E. Brener, S. S. Iyengar and J. R. Benton, Predictive Intelligence Military Tactical Analysis System (PIMTAS), a software system developed at Louisiana State University and the U.S. Army Topographic Engineering Center, 1999. Contact brener@bit.csc.lsu.edu for additional information.