ELSEVIER

Artificial Intelligence 95 (1997) 169-186

Artificial Intelligence

Research Note

# An event driven integration reasoning scheme for handling dynamic threats in an unstructured environment [1]

Yan Xia, S.S. Iyengar*, N.E. Brener

*Robotics Research Laboratory, Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803, USA*

Received May 1996; revised May 1997

**Abstract**

This paper presents an attempt to devise and develop a domain-independent reasoning system (DIRS) scheme for handling dynamic threats, and uses the scheme for automated route planning of military vehicles in an unstructured environment.

Automated route planning is a very important branch in applications of artificial intelligence. In a dynamic unstructured environment, instead of simply using static cost from a mobility model, a dynamic cost surface is constructed in which the total cost is a linear combination of the static cost and the dynamic cost.

The principal contributions of this paper are as follows: (i) A reasoning model called "DIRS" is proposed to quantitatively embed dynamic information, coordinate use of static and dynamic information, and handle real time events that happen outside the system. (ii) A temporal relation is applied in the route planning process for handling dynamic threats. (iii) Dempster-Shafer evidential theory is used to evaluate propagation of a dynamic threat. (iv) A detailed experimental analysis on automated route planning of military vehicles was conducted to study the performance of the DIRS model. © 1997 Elsevier Science B.V.

*Keywords:* Reasoning model; Dynamic threat; Route planning; Evidential theory; Autonomous robot

* Corresponding author. Email: iyengar@bit.csc.lsu.edu.

[1] This project was partly funded by US Army Research Office grant # DAAH04-93-G-0498 to Professor S.S. Iyengar.


*PII* S0004-3702(97)00035-0

## 1. Introduction

In everyday terms, planning means deciding on a course of action before acting. Here, in this paper, route planning for an object means determining an optimum path for the object before it begins moving. In different environments, for different objects, different factors should be taken into account to get an optimum path. For example, for a military vehicle, such as a tank or autonomous land vehicle, in a battlefield environment, the optimum path is determined by both time of travel and safety considerations. For a car, in an area where a football game just finishes, the football fans should be considered to figure out an optimum path to go through the area. For a robot that is planning to move to an indicated target in a dynamic area, static obstacles as well as the mobile obstacles in the area must be considered. In all of the above examples, we need to have a way to handle dynamic factors.

A threat, which is an object or event that prevents the actions of objects or events for which we are doing route planning, often owns both static and dynamic properties. The static properties are determined by the threat's static strength and the environment, and the dynamic properties are mainly determined by the mobility of the threat. For clarity, in the following sections, "object" only refers to that object for which we are doing route planning; otherwise, we call it a threat.

In this paper we present an event driven reasoning scheme for handling dynamic threats, and use the scheme for finding optimum paths for military vehicles in a dynamic battlefield environment. Here the dynamic battlefield environment means that there are mobile enemy forces like tanks, i.e. threats, which intend to destroy our object. Of course, our scheme can be used for civilian route planning like the examples we mention above.

This paper is organized as follows.

- First of all, we discuss related works on route planning and point out the reasons why these works are not adequate to handle dynamic threats.
- Secondly, a dynamic reasoning system is proposed to handle dynamic threats.
- Then, the proposed threat propagation method is discussed in detail.
- Finally, An experiment on a real terrain map illustrates how the reasoning in our system works and shows the effectiveness of our system in handling dynamic threats.

## 2. Related work

In order to improve the limitation of traditional production systems, an asynchronous production system (APS), which is capable of real time responses to unpredictable changes in the environment, has been described in [6] and [7]. This system provides a general control strategy for reasoning schemes. The architecture of the system is given in Fig. 1.

In Table 1, we outline other tactical planning systems that were proposed and point out the reasons why these systems are not adequate for handling dynamic threats in a dynamic environment.

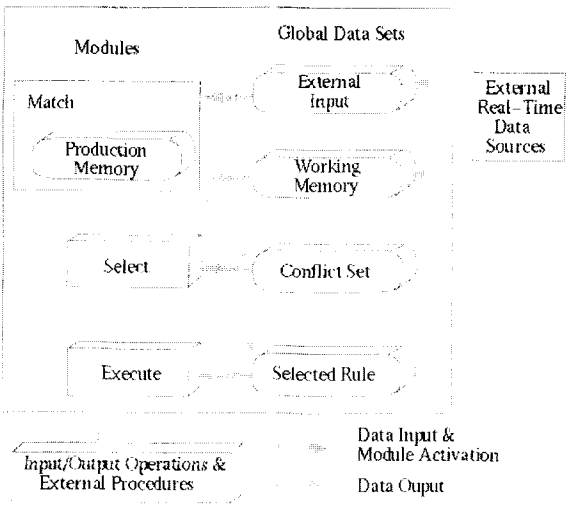


Fig. 1. Asynchronous production system.

Table 1
Comparison of related systems

| Investigator(s) | Brief description | Reason(s) system is not applicable to dynamic threats |
|---|---|---|
| J. Benton, S.S. Iyengar [2] (1995) | Two-level route planning in a battlefield environment | Cannot handle dynamic threats |
| G. Loberg [8] (1986) | A general planning approach | Not specifically for path planning. Used for large military unit like a corps |
| S. Badaloni [1] (1993) | A general planning approach | Used for hydraulic circuit maintenance |
| R.C. Strudeman [10] (1984) | Knowledge of threats is used for planning | Not specifically for path planning |
| W.B. Zavoli [12] (1985) | Finds a path based on a digital map database | The path is on-road only |
| W.R. Franklin [3] (1985) | An optimum path can be found for a robot | Not applicable to a battlefield environment |
| T.D. Garvey [5] (1987) | Knowledge of threats is used for planning optimum routes | Applicable only to helicopters. The condition of Dempster–Shafer evidential theory is not satisfied. |

## 3. A foundation of the integrated reasoning scheme for dynamic threats

### 3.1. Concepts and notations

All of the below concepts are based on graph $G : (V,E)$, where $V$ is a set of nodes and $E$ is a set of edges.

- A path is a sequence of edges $\langle n_0, n_1\rangle, \langle n_1, n_2\rangle, \langle n_2, n_3\rangle, \ldots, \langle n_{m-1}, n_m\rangle$, where $n_i \in V$, $\langle n_i, n_{i+1}\rangle \in E$, $0 \leqslant i < m$ and $n_i \neq n_j$ if $i \neq j$, $0 \leqslant i, j \leqslant m$. We call the above path a path between node $n_0$ and node $n_m$.
- The mobility weight, $mw(i,j)$, is used to measure the mobility on a path between node $i$ and node $j$.
- The static threat weight, $stw(i,j)$, is used to measure the static threat on a path between node $i$ and node $j$.
- The static weight, $sw(i,j)$, is a combination of the mobility weight and the static threat weight on a path between node $i$ and node $j$.
- The dynamic threat weight, $dw_i(x)$, is used to measure the dynamic threat of threat $T_i$ on graph node $x$.
- The path weight, $pw(i,j)$, is a combination of static weight $sw(i,j)$ on a path between node $i$ and node $j$, and dynamic weights which associate with the nodes on the path.
- $T$ is a set of threats, i.e. $T = \{T_i \mid 1 \leqslant i \leqslant n\}$. $T(x)$ is the combination of the influence of all dynamic threats on node $x$ and $T_i(x)$ is the influence of dynamic threat $T_i$ on node $x$.

### 3.2. A computational architecture of reasoning scheme

#### 3.2.1. Outline of scheme

The architecture of our reasoning system is shown in Fig. 2.

Here, inputs are an unstructured environment in which we will do route planning, information of threats which are dynamic in the environment, a starting position and an ending position in the environment. Output will be one or more optimum paths between the two positions.

When all data are inputted to the system, the first thing that is done is generate a weighted graph from the environment with threat information and the two positions. The graph should satisfy the following conditions:

- For any two adjacent graph nodes, the locations of the nodes are connected.
  Note that when we say two points are connected, this means that there is a path which links the two points in the environment.
- For any point, there is at least one graph node whose location in the environment is connected with that point. We call the node that point's neighbor node.
- For any two points which are connected, either there is at least one path which links their neighbor nodes in the graph or both of them have a same neighbor node.
- The starting and ending points are the two graph nodes' location.
- Threat's locations may never be any graph node's location.

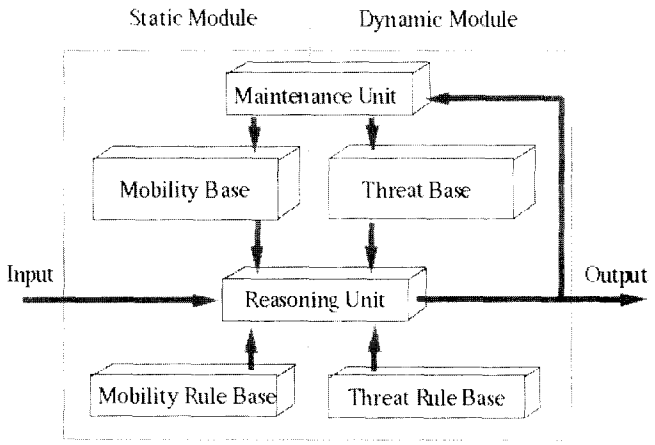


Fig. 2. Reasoning system for dynamic threat.

The graph we get is the representation space we will work on. The mobility weight and static weight, which are associated with edges in the graph, are calculated based on the environment and threats' information. The ways to calculate the mobility weights and static threat weight are sensitive case by case. We will present a way in Section 5.1 on automated route planning of military vehicles.

*3.2.2. Static module*

In the mobility base (MB), we store the graph which is generated above. For edge $\langle x, y\rangle$, which associates with static weight $sw(x, y)$, we use the following format to represent it:

$$P(x, y) : sw$$

where $sw \in \mathbb{R}^+$, $\mathbb{R}^+$ is the set of positive real numbers. This means that there is a path between adjacent nodes $x$ and $y$ with static weight $sw$.

In fact, we use two integers, a mobility factor weight ($MFW$) and a static threat factor weight ($STFW$), to balance the influence of mobility and static threats. The static weight $sw$ is:

$$sw = W(mw, stw) = MFW * mw + STFW * stw.$$

In the mobility rule base(MRB), the mobility reasoning rules are stored as follows:

$$P(x, y) : w \iff P(y, x) : w$$

This rule means that any path between two nodes is a two-way path.

$$P(x,y) : w_1 \wedge P(y,z) : w_2 \implies P(x,z) : w_1 + w_2$$

This rule means that the weight on a path is equal to the sum of the weights which are on the subpaths of the path. The path consists of the subpaths, none of which overlap nor overcross each other.

$$P(x,y) : w_1 \wedge P(x,y) : w_2 \implies P(x,y) : \min\{w_1, w_2\}$$

This rule means that if there are two paths between any two nodes, we will choose the one with the smaller weight.

*3.2.3. Dynamic module*

In this module, we propagate the threats to get dynamic weights for the graph nodes. The method of threat propagation will be discussed in Section 4.

Due to dynamic reasons, we consider not only the dynamic weight for a node, but also the time that threat moves to the node. Therefore, in the threat base, we will store the above information as follows:

$$T_i(x) : t_i, dw_i$$

This means that threat $T_i$ may propagate to node $x$ in time $t_i$ with dynamic weight $dw_i$, $dw_i \in \mathbb{R}^+$.

For every node $x$,

$$T(x) = \{T_i(x) : t_i, dw_i \mid T_i \in T\}$$

In the threat rule base (TRB), we suppose that $y_1, y_2, \ldots, y_n$ $(= y)$ be all nodes on the path from node $x$ to node $y$ and $t_1, t_2, \ldots, t_m$ are all nodes, each of which a threat locates on currently. The following rules will be applied:

$$(P(x,y) : sw) \wedge T(y_1) \wedge T(y_2) \wedge \cdots \wedge T(y_n) \implies P(x,y) : pw$$

where

$$sw = W(mw, stw), \qquad pw = W'(sw, dw),$$

$$dw = \sum_{i=1}^{n} dw(y_i) \quad \text{and} \quad dw(y_i) = \sum_{j=1}^{m} dw_j(y_i)$$

$$\text{if } time(mw(x, y_i)) > time(mw(t_j, y_i)).$$

The function $W'(sw, dw)$, which combines $sw$ and $dw$ using the static factor $weight(SFW)$ and dynamic factor $weight(DFW)$, is similar to the function $W$ which combines $mw$ and $stw$.

The function $time(mw(x,y))$ is used to calculate the time which is needed to traverse the path with mobility weight $mw(x,y)$. As we know, the function $time(mw)$ should be a monotone increasing function, that is, the larger the mobility weight is, the more time is needed to travel the path. Thus the following relationship must be satisfied:

$$time(mw_1) > time(mw_2) \iff mw_1 > mw_2$$

Because we do not really care the exact amount of time which threats or the object spends on the way, what we really care is only the relation between the amount of time which threats spend on the way and the amount of time the object spends on the way. Therefore, we can use $mw$ to replace the value of $time(mw)$, that is $time(mw) = mw$. Here we suppose that the mobility of all threats and objects are the same.

Let us take a detailed look at computing $dw(y_i)$. For clarity, we rewrite the formula as follows:

$$dw(y_i) = \sum_{j=1}^{m} dw_j(y_i) \quad \text{if } time(mw(x, y_i)) > time(mw(t_j, y_i)).$$

The condition $time(mw(x, y_i)) > time(mw(t_j, y_i))$ determines whether or not threat $T_j$ will influence the planned path from node $x$ through node $y_i$. If time $time(mw(x, y_i))$, which is needed for the object to reach node $y_i$, is greater than time $time(mw(t_j, y_i))$, which is needed for threat $T_j$ to reach node $y_i$, the dynamic weight $dw(y_i)$ will be considered. Otherwise, threat $T_j$ will be ignored on node $y_i$ for the current planned path. This is the temporal relation we use in our reasoning scheme.

#### *3.2.4. Reasoning and maintenance modules*

The reasoning module is called for all reasoning. For example, when the maintenance module propagates a threat, the module is called to find an optimum path which only depends on mobility without threat in order to move the threat.

In the maintenance module, once the situation of threats is changed (a new threat is found, a threat disappears, or a threat has moved), a change to the graph on which we are planning a path is activated, in order to make the graph reflect the current realistic situation. Then recalculation on static and dynamic threats is also motivated by the graph's change. Eventually, new optimum paths are planned in the new situation.

## 4. Dynamic threat propagation

### *4.1. Threat propagation on tree*

In fact, dynamic threat propagation is a quantitative predication on a threat's move intention.

In order to move threats along with graph, we now introduce a new graph node for each threat. This node's location in the environment is the same with threat's location. Also the node is linked to some closed graph nodes whose locations in the environment are connected with the new node's location. Here we get a new graph. Note that this new graph is only used for threat propagation.

Now let us take a look to threat propagation on a graph. First of all, no matter which path the threat probably moves along with, it will start from its initial node. So, all paths along which the threat will probably move form a tree in which the initial node is root. We call this tree a threat tree. Note that a weight on a threat tree is only mobility weight. Therefore, the threat propagation is discussed based upon threat tree.

For clarity of discussion, we assume the strength of the threat at the root is 1 and the threat we are discussing is $T_i$. Let

$$LEAF = \{x \mid x \text{ is a leaf of the threat tree}\}, \qquad n = |LEAF|,$$
$$TREE = \{x \mid x \text{ is a node of threat tree}\}.$$

The method of how to calculate the influence of a threat on the tree's leaf nodes is based on the following assumptions.

- The whole threat will move together. This means, after the threat moves, nothing will be left at the original site.
- The influence of a threat to every leaf node on the corresponding threat tree is only based on the mobility from the root to the leaf node.

So we get

$$\sum_{j=1}^{n} dw_i(x) = 1 \quad \text{and}$$
$$dw_i(x) = (mw - mw(x))/(n-1) * mw, \quad \text{where } mw = \sum mw(x),\ x \in LEAF.$$

Here $mw(x)$ is the mobility weight of the path from the root to leaf $x$.

For non-leaf nodes $x \in TREE - LEAF$ on the tree, we think the dynamic weight $dw_i(x)$ should be equal to the sum of the dynamic weights of all sons of node $x$. So, we have

$$dw_i(x) = \sum_{j=1}^{m} dw_i(y_j), \quad x \in TREE - LEAF,$$

where $y_j$ is a son of node $x$ and $m$ is the number of sons of node $x$.

From the discussion above, we obtain a method to calculate the dynamic threat of every node on the threat tree. In fact, this method is an application of the Dempster–Shafer evidential theory [9]. Here our domain is *LEAF*. The method to acquire the value of dynamic threats on leaves corresponds to the basic probability assignment in the evidential theory. And we use the belief function of the evidential theory to calculate the dynamic threats of the non-leaf nodes. All conditions of the Dempster–Shafer evidential theory are satisfied in this application.

### 4.2. Extended threat propagation

For the other nodes outside the indicated tree, we only consider those nodes which are adjacent to one of the tree nodes except the root. The reasons of excluding the root here are as follows:

- We assume that the threat's initial intention to move is apparent. The probability of moving back is very small.
- When we handle the static threat, the adjacent area has been considered.

The influence of a threat to the non-tree nodes should be inversely proportional to the mobility weight which the threat will encounter as it moves along the path. Therefore the method to calculate the influence of a threat for a non-tree node $x$ is:

$$dw_i(x) = (mw(y)/mw(y) + mw(x,y)) * dw_i(y),$$

where $mw(y)$ is the mobility weight from the root to tree node $y$ which is adjacent to node $x$ and $mw(x,y)$ is the mobility weight of the edge between node $x$ and tree node $y$.

For a given threat, one non-tree node may get more than one dynamic weight because the node may be adjacent to more than one tree node. So, we have to choose only one from all of the dynamic weights. The method of selection is as follows:

- First get the path(s) with the least mobility weight from the root to the node.
- If more than one path with the same mobility weight is obtained from the above rule, we choose the path with the maximum dynamic weight.

### 4.3. Algorithm

Threat propagation algorithm in our system consists of two steps. The first one, called the top down step, calculates the time which the threat needs to reach the corresponding node from the root. The other, which is called the bottom up step, calculates the threat propagation, that is, the dynamic weight on every node in threat tree.

The algorithm of threat propagation is as follows:

**Input:** A graph and all threats in $T$, each of which has a threat tree.
**Output:** Dynamic weight for every node on the graph.
**Begin**
1. $i = 1$;
2. **forall** $(T_i \in T)$ **do** {
3. **forall** $(x_j \in LEAF_i)$ **do** {
4. **calculate** $mw(x_j)$;
which represents the mobility from root to $x_j$.
}
5. $mw = \sum_{j=1}^{n_i} mw(x_j)$; where $n_i = |LEAF_i|$.
6. **forall** $(x_j \in LEAF)$ **do** {
7. $dw_i(x_j) = (mw - mw(x_j))/(n_i - 1)mw$.
}
8. **forall** $(x_j \in TREE - LEAF)$ **do** {
9. $dw_i(x_j) = \sum_{t=1}^{m} dw_i(y_t)$,
where $y_t$ is a son of $y$ in $T_i$ and $m$ is the number of the sons.
}
10. **forall** $(x_j \in TREE$ and $x_j \neq$ root) **do** {
11. **forall** ($y$ is linked to $x_j$, where $y \overline{\in} TREE$) {
12. $dw_i(y) = (mw(x_j)/mw(x_j) + mw(x_j, y)) * dw_i(x_j)$
where $mw(x_j, y)$ is the mobility between $x_j$ and $y$.
}
}
}
**End**

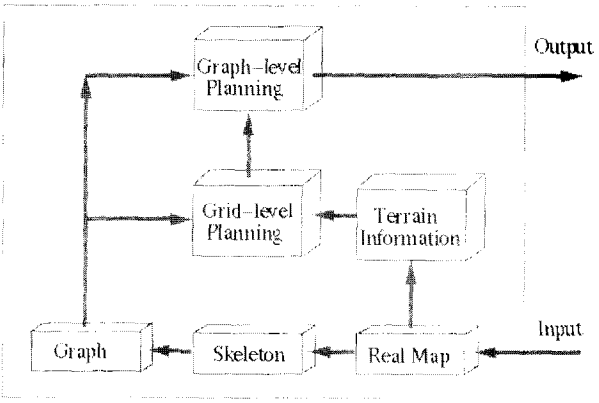


Fig. 3. PIMTAS architecture.

## 5. Experiments on real map

### 5.1. PIMTAS

A United States Army project, Predictive Intelligence Military Tactical Analysis System (PIMTAS), is ongoing at the LSU Robotics Research Laboratory and the United States Army Topographic Engineering Center. Benton et al. [2] proposed a method called two-level hierarchical route planning in order to avoid combinatorial explosions. Here we describe briefly the PIMTAS system. The system architecture is shown in Fig. 3.

A mobility map in which every pixel has a value of either 1 or 0, which represent GO and NO_GO mobility respectively, is called a binary map. A thinning algorithm [2] can be applied to a binary map in order to generate a skeletal structure that will correspond to a line drawn along the center of mobility corridors. A graph can be generated from the skeleton. An example is shown in Fig. 4, in which the left panel is a real binary map and the right panel is its corresponding skeleton.

In PIMTAS, the grid level route planner is used to generate the precise route between adjacent graph nodes by searching pixel by pixel on the binary map, and the graph level route planner is used to efficiently plan a route over a larger area by searching node by node on the graph acquired from the skeleton of the map.

When a threat is introduced in a map, a small area around the threat becomes a NO_GO area because it is too dangerous to go through the area. A larger area around the NO_GO area, which is influenced by the threat directly, is called static threat area. The dynamic threat's influence will be beyond the static threat area.

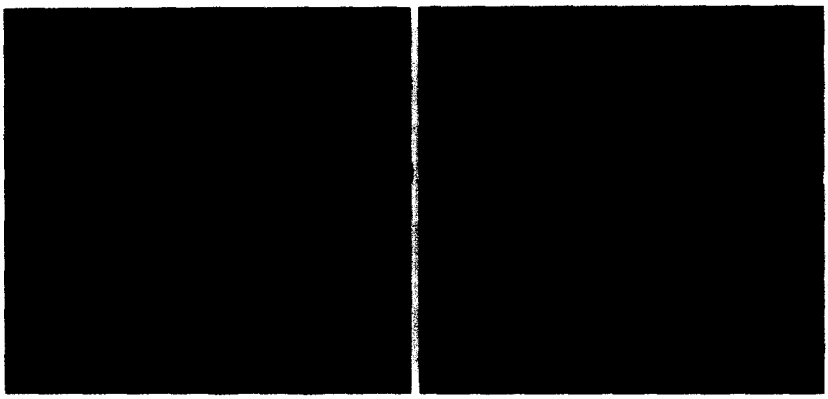

Fig. 4. A real map and its skeleton.

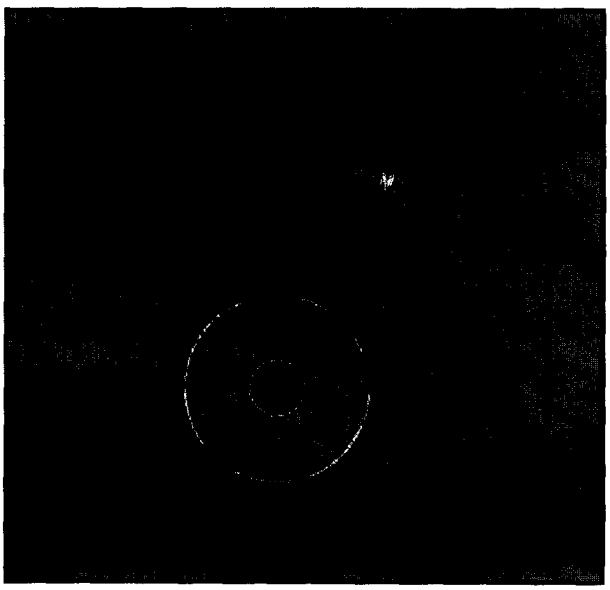

Fig. 5. A real terrain map with a threat.

In Fig. 5, we present a real terrain map with a threat which produces a threat NO_GO area modeled by the smaller circle, a static threat area in between the two circles, and a dynamic threat outside the larger circle.

In the map, every pixel has a mobility value. But only in the static threat area is every pixel assigned a static threat value according to the type of threat and the distance from the pixel to the threat center. When a grid level planner searches for an optimum path between two adjacent graph nodes, the mobility weight and static threat weight

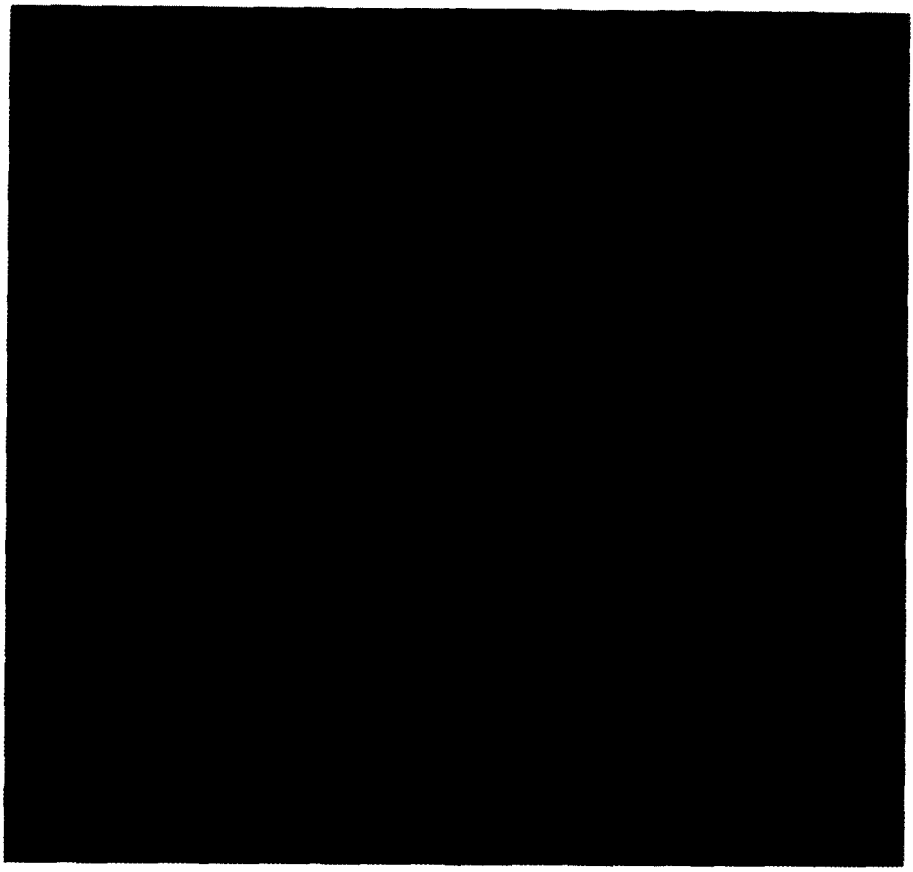

Fig. 6. A graph from a real map without threats.

associated with the path are respectively equal to the sums of the mobility and static threat values of all of the pixels on the path. Therefore, we can get the static weight of a graph edge by combining the mobility weight and static threat weight of the corresponding grid level path.

### 5.2. Analysis

We now show the results that we get when the method proposed in Sections 3 and 4 is embedded in PIMTAS.

Now we assume that $MFW = 1$, $STFW = 1$, $SFW = 1$ and $DFW = 1$. The combination function which calculates the path weight is as follows:

$$SFW * (MFW * mw + STFW * stw) + DFW * dw.$$

First of all, we obtain a graph, shown in Fig. 6, from the skeleton, which is shown in Fig. 4. The indicated mission is to find two optimum, non-competing paths from graph

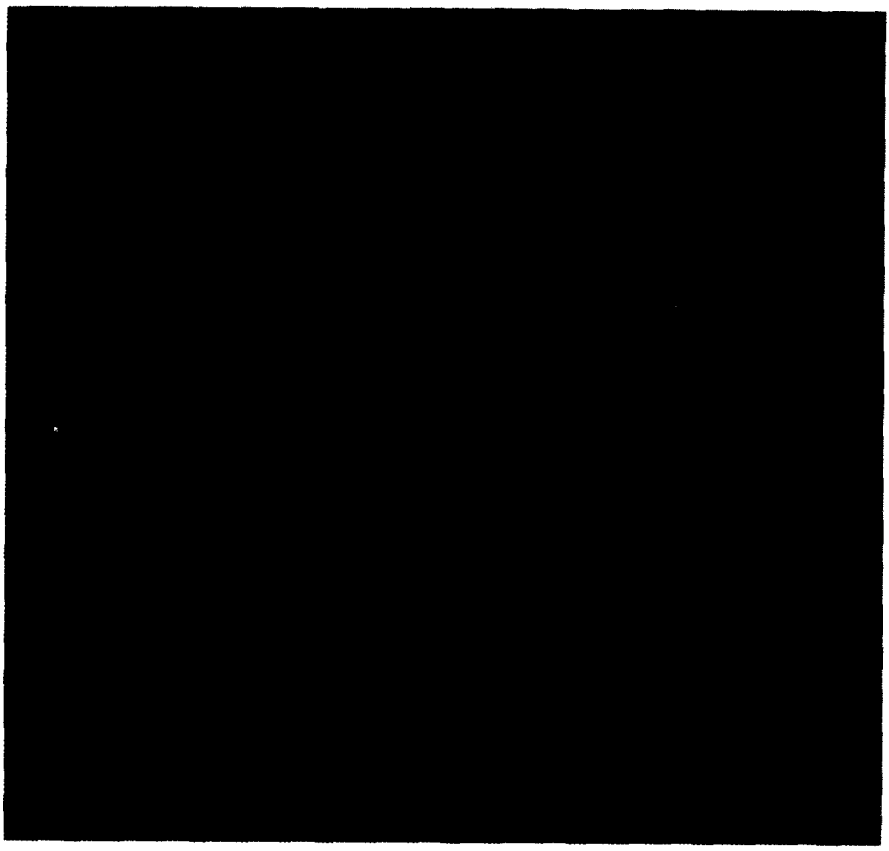

Fig. 7. Two paths between two given points without threats on the map.

node 85 (starting node) to node 22 (destination node). In Fig. 7, the two paths found go through the following nodes:

- Path 1: 85–92–12–13–15–16–109–20–21–22, with weight 342.
- Path 2: 85–86–93–91–9–8–7–6–33–30–29–23–22, with weight 490.

Now we add a threat near node 40 in the lower right corner of Fig. 6. This threat creates a NO_GO area modeled by a circle, as shown in Fig. 9. We re-thin the map to get a new skeleton, which is shown in Fig. 8. We then use the new skeleton to obtain a new graph, shown in Fig. 9. The graphs in Figs. 6 and 9 are almost the same except for the part of the graphs near the threat location and the numbers assigned to the nodes.

We indicate a threat tree with root $T$ as shown in Fig. 10, where a new node $T$, which corresponds to the threat position, is added to the graph shown in Fig. 9 for threat propagation. The other nodes in the tree are those nodes from Fig. 9 with the same labels.

Now, from the algorithm in Section 4.3, we get the mobility weight from the threat to the leaf nodes and the dynamic weights of the leaf nodes, which are shown in Table 2.

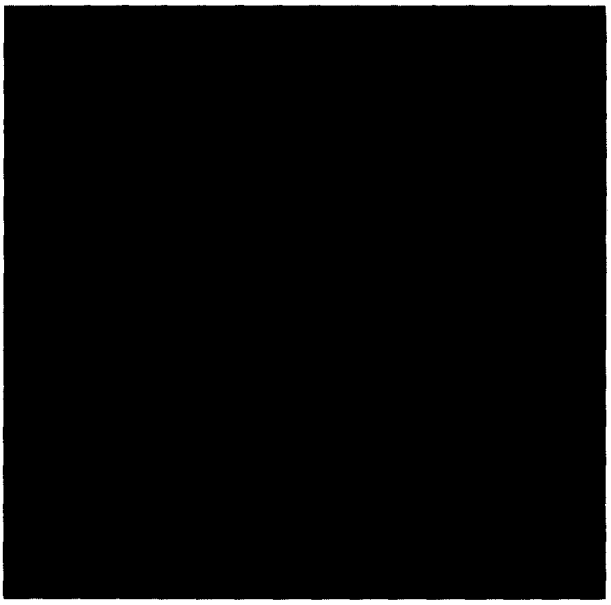

Fig. 8. New skeletons.

Table 2
Weights of leaf nodes

| Node number | 7 | 33 | 30 | 44 | 29 |
|---|---|---|---|---|---|
| Mobility weight | 141 | 161 | 138 | 68 | 113 |
| Dynamic weight | 19.3 | 18.5 | 19.4 | 22.3 | 20.5 |

Table 3
Dynamic weights of non-leaf nodes

| Node number | 6 | 32 | 31 | 28 | 43 |
|---|---|---|---|---|---|
| Dynamic weight | 37.8 | 57.2 | 19.4 | 20.5 | 42.8 |

Table 4
Weights of some non-tree nodes

| Node number | 8 | 36 | 35 | 23 |
|---|---|---|---|---|
| Mobility weight | 214 | 212 | 292 | 156 |
| Dynamic weight | 12.7 | 12.8 | 10.2 | 14.8 |

Also, we get the dynamic weights for the non-leaf nodes, shown in Table 3.

In Table 4, we give the mobility weights of some relevant nodes which may appear on the planned path and their dynamic weights.

Among the nodes given in Tables 2–4, the ones which appear in path 2 are nodes 8, 7, 6, 33, 30, 29 and 23, and thus the total dynamic weight of this path is $12.7 + 19.3 +$

Fig. 9. A graph from a real map with threats.

$37.8 + 18.5 + 19.4 + 20.5 + 14.8 = 143$. So, after the threat is added, the weight of path 2 increases to 633.

As we mentioned previously, our system also considers the time factor in deciding whether or not a dynamic threat on a node will be considered when we plan a path through the node. In Table 5, we list the amount of time that the threat needs to travel to a node, and the amount of time that the object needs to travel to the node. Here we find that the object time to node 8 is less than the threat time to the same node. According to the discussion in Section 3.2.3, the dynamic weight on node 8 should be ignored. Therefore, the dynamic weight of path 2 is 130 and the weight of the path increases to $130 + 490 = 610$.

Now let us set $DFW = 2$ instead of 1. This means that the dynamic weight of path 2 will be $2 * 143 = 286$. After ignoring the dynamic weight on node 8, we get the dynamic weight of the path to be $2 * 130 = 260$. Therefore, the weight of the path is $260 + 490 = 750$.

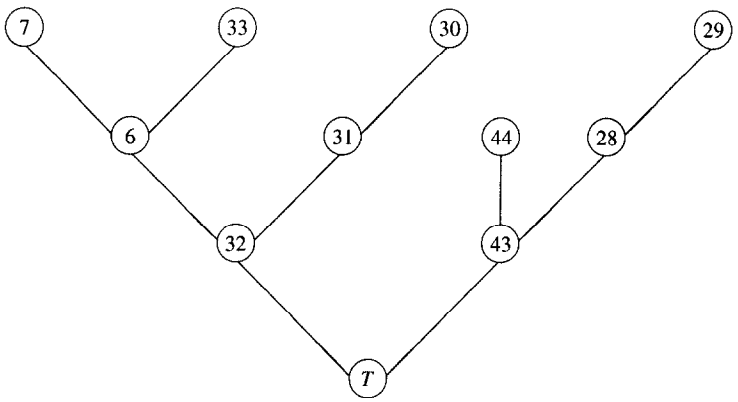


Fig. 10. A threat tree.

Table 5
Mobility weights of some nodes due to travel by a threat and travel by the object from the starting node

| | Threat | Object |
|---|---|---|
| node 8 | 214 | 147 |
| node 7 | 141 | 220 |
| node 6 | 103 | 258 |
| node 33 | 161 | 316 |
| node 30 | 138 | 349 |
| node 29 | 113 | 404 |
| node 23 | 156 | 447 |

Fig. 11 shows the result of recalculating path 1 and 2 after the threat has been added. Here the two paths go through the following nodes:

- Path 1: 82–89–12–13–15–16–106–20–21–22, with weight 342.
- Path 2: 82–81–71–70–66–65–64–63–62–57–55–46–47–49–22, with weight 701.

In contrast to the two paths in Fig. 7, path 2 in Fig. 11 is totally different due to the influence of the threat, which forces path 2 far away from the threat. Thus the new path 2 is much longer than the old one. The reasoning behind choosing this longer path as the new path 2 is that the weight (701) associated with the longer one is less than the new weight (750) associated with the original path 2.

From the discussion of the above example, the results achieved from our system describe the threat propagation and the influence of the threat on path planning. For example, although the threat propagates to node 8, when the path through node 8 is to be planned, the influence of the threat is discarded because the time factor is also considered in our system. This result demonstrates our system's effectiveness in handling

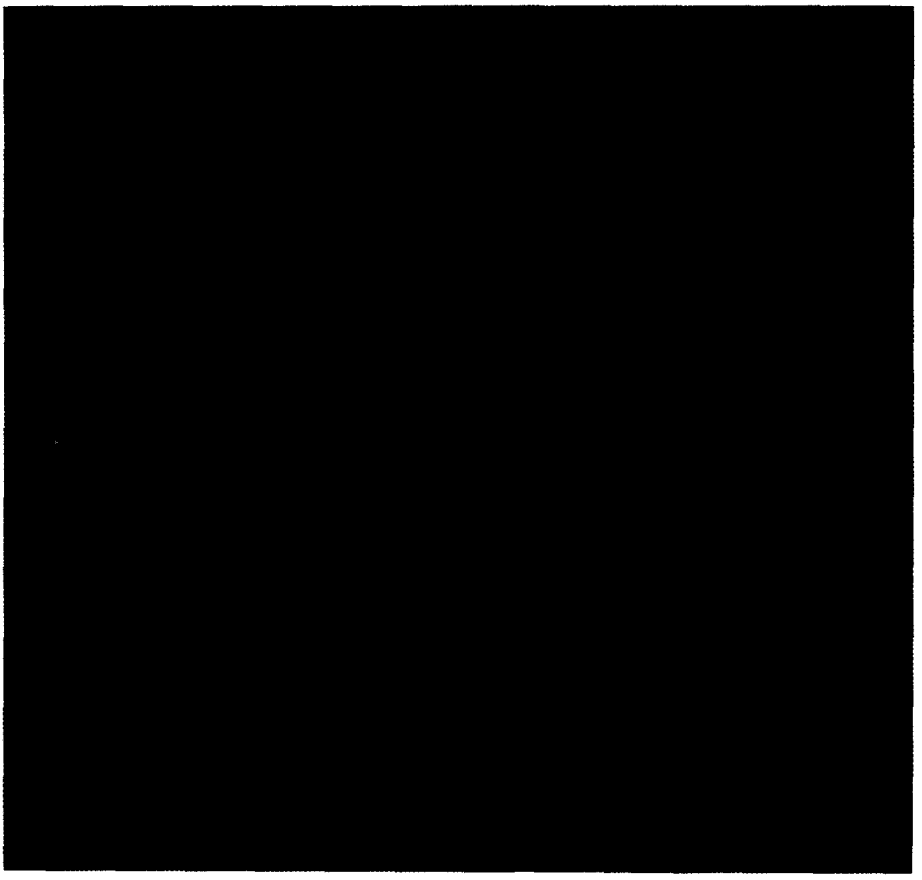

Fig. 11. Two paths between two given points with threats on the map.

## 6. Conclusion

In the discussion above, a dynamic reasoning model has been described. The model, in general, is applicable to any dynamic environment. We have then used this model to build up a reasoning system for handling dynamic threats in a battlefield environment. The results from PIMTAS have demonstrated the effectiveness and efficiency of our system. For the reasoning, we not only have applied a temporal relation, but also have given an application of Dempster–Shafer evidential theory.

## Acknowledgments

We acknowledge the comments of the reviewers which improved the presentation of this paper. This work was supported in part by the United States Army Topographic

Engineering Center. We acknowledge special thanks to Dr. John Benton for his useful technical suggestions. Also we would like to thank Dr Weian Deng and Paul Clavier who worked in the project.

## References

[1] S. Badaloni, E. Pagello, L. Stocchiero and A. Zanardi, Making an autonomous robot plan temporally constrained maintenance operations, in: *Advances in Artificial Intelligence, Proceedings 3rd Congress of the Italian Association for Artificial Intelligence* (1993) 290–301.

[2] J. Benton, S.S. Iyengar, W. Deng, N. Brener and V.S. Subrahmanian, Tactical route planning: new algorithm for decomposing the map, *Internat. J. Artif. Intell. Tools Appl.* 5 (1&2) (1996) 199–218.

[3] W.R. Franklin, V. Akman and C. Verrilli, Voronoi diagrams with barriers and on polyhedra for minimal path planning, in: *The Visual Computer* (Springer, Berlin, 1985) 133–150.

[4] R.M. Fung and C.Y. Chong, Metaprobability and Dempster–Shafer in evidential reasoning, in: L.N. Kanal and J.F. Lemmer, eds., *Uncertainty in Artificial Intelligence* (Elsevier Science, Amsterdam, 1986) 295–302.

[5] T.D. Garvey and L.P. Wesley, Knowledge-based helicopter route-planning, in: *The DARPA Workshop*, Austin, TX, 1987.

[6] S.S. Iyengar, W. Deng and A.A. Nanavati, Dynamization of event based production system, in: *Proceedings Knowledge Based Computer Systems*, Poona, India (1990).

[7] S.S. Iyengar, J. Graham, V.G. Hegde and P. Graham, A concurrent control architecture for autonomous mobile robots using asynchronous production systems, *Automation in Construction* 1 (1993) 371–401.

[8] G. Loberg and G.M. Powell, Operational military planning: an artificial intelligence approach, in: *Artificial Intelligence for Military Applications* (Operations Research Society of America, 1986) 27–55.

[9] S.J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach* (Prentice-Hall, Englewood Cliffs, NJ, 1995) Chapter 12, p. 462.

[10] R.C. Strudeman, A knowledge acquisition system for threat assessment, mission area analysis and combat development planning, in: *Proceedings Artificial Intelligence and Robotics Symposium* (1984).

[11] Y. Xia, A reasoning system for handling dynamic threat in an unstructured environment, MS Thesis, Louisiana State University, Baton Rouge, LA (1996).

[12] W.B. Zavoli, S.K. Honey and M.S. White, A land vehicle navigation system supported by a digital map data base, in: *Proceedings Artificial Intelligence and Robotics Symposium* (1985).