INFORMATICA, 2006, Vol. 17, No. 1, 95–110

Scheduling Trajectories on a Planar Surface with Moving Obstacles

Emmanuel STEFANAKIS
Spade Team, Institut Autonome Intelligente Systeme
Fraunhofer-Gesellschaft, Sankt Augustin, Germany
e-mail: stefanak@dbnet.ece.ntua.gr

Received: September 2004

Abstract. An algorithm for scheduling the trajectory of a point object, which moves on a plane surface comprising a set of moving obstacles, is introduced. Different quantitative criteria may be met by the schedule, e.g., the course connecting two individual locations being the shortest in length, the least expensive, the fastest as regard to its duration, etc. A prototype system that implements the algorithm is presented. Several example scenarios are also discussed.

Key words: spatio-temporal modeling, graphs, trajectory schedule, optimum paths, moving objects.

1. Introduction

The *scheduling of an object (e.g., a vessel) trajectory* is a common problem in human navigation and appears very often in applications such as Cartography, Logistics, Robotics and Geographic Information Systems (GIS). Moving between two physical locations can be, basically, accomplished based on various alternative schedules. Each schedule can be characterized and quantitatively described by some objective criteria. For instance, we may look for – to name a few:

- ✓ the shortest, longest, fastest, or least expensive trajectory connecting the two locations,
- ✓ a trajectory that departs from the start location at time t_s, and arrives at destination at time t_d,
- ✓ a trajectory, which further adds to the previous one the constraint to cross an intermediate location at time t_i and reside there for the time interval $[t_i, t_j]$.

Obviously, there are three additional *parameters* that should be clarified, before we browse for the trajectory that meets the criteria above. These describe (a) the *dimensions of the space* where movement takes place, (b) the *constraints of movement*, and (c) the *dynamic nature of space* in time.

As for the space, all intermediate locations that compose the trajectory (including start and destination locations) belong to a space that may have – depending on the application – one, two, two and a half (for curved surfaces, like the earth) or three dimensions

in general. In this study, we limit the discussion on movements in a two-dimensional (plane) surface. All concepts can be readily extended and applied to spaces of higher dimensionality (Stefanakis and Kavouras, 1995; Styliadis *et al.*, 2003).

As for the constraints of movement, the trajectory connecting two physical locations may be limited to the chains of an existing linear network or not. In the former case, graph theory can be applied to simulate the movement (Johnson, 1977; Gibbons, 1985; Sedgewick, 1990; Rich and Knight, 1991; Russell and Norvig, 1995). In the latter case, where the movement is not confined to a linear network, existing raster-based (Warntz, 1961; Lindgren, 1967; Goodchild, 1977; Church *et al.*, 1992; Van Bemmelen *et al.*, 1993; Douglas, 1994ab) or vector-based (Mitchel and Papadimitriou, 1991) approaches may be applied. In this study, we examine trajectories in space, and we apply an approach, recently introduced by Stefanakis and Kavouras (1995, 2002). This algorithm is based on the degeneration of the space under study into a network, which can be simulated by a weighted graph, so that algorithms of graph theory and artificial intelligence can be easily adopted to indicate the optimum path(s) for the desired trip.

Finally, as for the dynamic nature of space in time, there are two alternatives. In the first alternative, the space is static, in the sense that the cost of movement per unit of movement (e.g., one meter) remains unchanged over time everywhere in space. In the second alternative, the cost of movement changes over time. The cost of movement is described through a (spatial) *cost model* (Stefanakis and Kavouras, 2002). Obviously, time is a parameter of the cost model. In this study, we examine a simple scenario, where the cost model applies the function of *Euclidean distance*. That is, the cost of movement (c_{AB}) from a location $A(x_A, y_A)$ to a location $B(x_B, y_B)$ is equal to

$$c_{AB} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}.$$

Additionally, we assume that the space comprises a set of *moving obstacles*, which constraint the access to specific regions (covered by the obstacles) during specific temporal intervals.

A real world application that may be supported by the configuration of our study is the scheduling of the sea course for a boat in a relatively small region of the earth (in order to satisfy the condition of planarity). The course is constrained by a set of static obstacles, i.e., the islands and continents; and by a set of moving obstacles, i.e., the other vessels.

The discussion is organized as follows. Section 2 describes the algorithm for scheduling the trajectory of an object in a dynamic space with obstacles. Section 3 presents a prototype system which implements the algorithm and several example scenarios generated by the system. Finally, Section 4 concludes the discussion and proposes some hints for future research.

2. The Algorithm

Assume a two-dimensional plane surface S (Fig. 1). For simplicity reasons, the surface is orthogonal – with its borders parallel to the X, Y-axes – and described through two

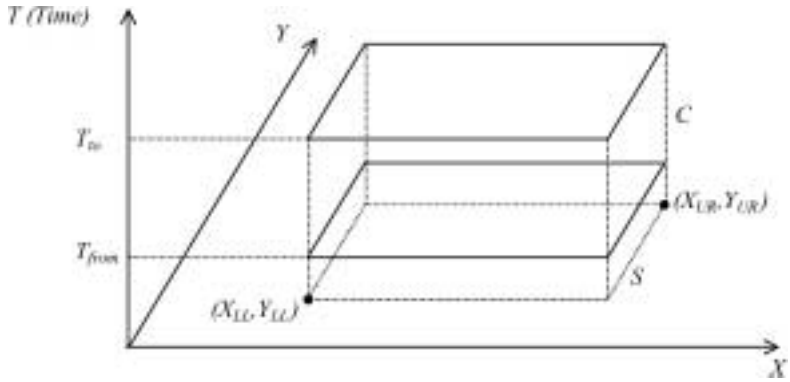

Fig. 1. The space-time.

pairs of (x, y) coordinates, the lower left (or south west – X_{LL}, Y_{LL}) and the upper right (or north east – X_{UR}, Y_{UR}) corners. The space is considered during a temporal interval $[T_{from}, T_{to}]$, defined by a pair of time instances, the T_{from} and T_{to}, where T_{to} is subsequent to T_{from}. We call this period of time as *space life*. Hence, a *spatio-temporal cube C* defined by the triples (X_{LL}, Y_{LL}, T_{from}) and (X_{UR}, Y_{UR}, T_{to}) is considered.

The surface S comprises a set of moving obstacles (MO). In other words, a set of objects, which are moving in the spatio-temporal cube C. Each obstacle MO_i has a circular shape with a radius r_i, and carries out a straight route with a constant velocity v_i. Specifically, each obstacle MO_i is defined by the following set of parameters:

$$(r_i, x_{from-i}, y_{from-i}, t_{from-i}, x_{to-i}, y_{to-i}, t_{to-i}),$$

where the triples $(x_{from-i}, y_{from-i}, t_{from-i})$ and $(x_{to-i}, y_{to-i}, t_{to-i})$ correspond to the starting and ending locations of the obstacle MO_i in space-time.

Fig. 2 presents an example obstacle MO_i with radius r_i on surface S (a projective view), which travels from point $A(x_A, y_A)$ to point $B(x_B, y_B)$, during the temporal interval $[t_{A-i}, t_{B-i}]$. The object velocity is constant and equal to

$$v_i = \frac{\sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}}{(t_{B-i} - t_{A-i})}.$$

What we are looking for is the schedule (if any) of a point object, which moves on the surface S and its course: (a) connects two specific locations in space, (b) falls inside the spatio-temporal cube C, (c) satisfies the schedule criteria, and (d) does not hit any moving object.

In order to accomplish this task, we adopt the approach introduced by Stefanakis and Kavouras (1995, 2002) – which supports the navigation in a static space based on quantitative criteria – and extend it here so that it may be applicable to a dynamic space. As stated previously, the discussion is limited to a plane surface, which comprises a set of moving obstacles of circular shape. The schedule refers to a point object, which moves

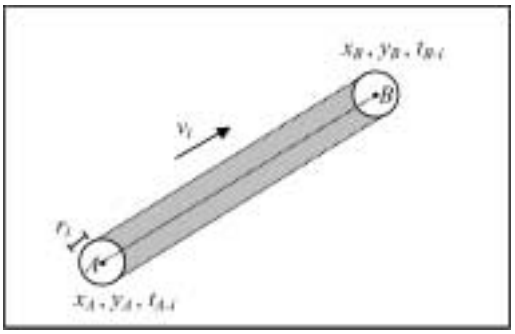

Fig. 2. An example of a moving obstacle.

on this surface and does not hit any moving obstacle at any time. Notice that obstacles size may be enlarged appropriately to include the moving object size – if the latter is not a point object – and/or the security distance (buffer zone) between the point object and the obstacles themselves (to avoid collision).

The algorithm consists of the five steps, which are described in the following Subsections:

1. Establishment of a network in space.
2. Formulation of the travel cost model.
3. Computation of the temporal intervals during which nodes and edges are not accessible.
4. Solving the network.
5. Determination of the schedule.

2.1. *Establishment of a Network in Space*

The inconvenience of movement in space is the infinite number of spots (i.e., point locations or nodes), involved in the determination of a path. The proposed solution (Stefanakis and Kavouras 1995, 2002) to overcome this problem is based on the technique of discretization of space. *Discretization* (Laurini and Thompson, 1992; Worboys, 1995) is the process of partitioning the continuous space into a finite number of disjoint areas or volumes (cells), whose union results in the space. By representing each of these cells with one node (e.g., its center point), a finite set of nodes is generated.

Obviously, the number of nodes depends on the size of the cell. If these nodes are interconnected through edges, a linear network is established, and appropriate algorithms available in graph theory and artificial intelligence can be applied to support the navigation. How nodes are interconnected is related to the degrees of freedom characterizing the movement. In this study we adopt a common scheme, which is based on the *regular grid* tessellation. More details can be found in (Stefanakis and Kavouras, 2002).

Specifically, a regular grid is superimposed on the plane surface. A network node is then located on the centroid of each cell (Fig. 3). Then, network edges are established to connect the network nodes. These edges are driven from the regular grid as follows. Each

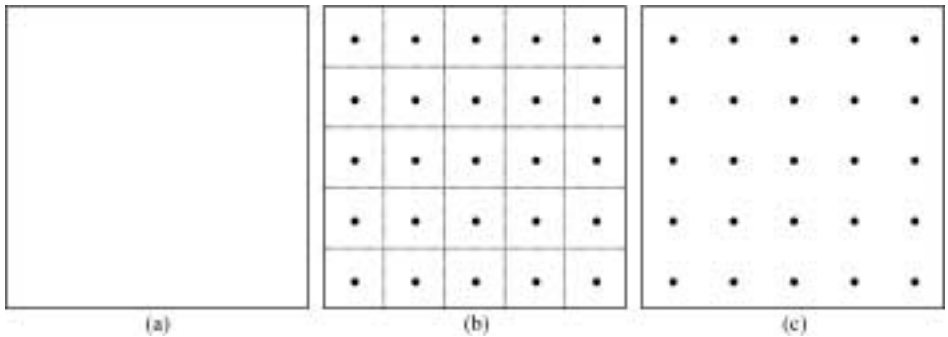

Fig. 3. Establishing the network nodes. The space (a), the tessellation superimposed on the space (b), and the resulting nodes (c).

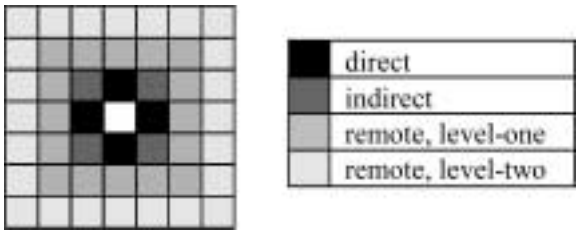

Fig. 4. Types of cell neighbors in a regular grid.

cell has three types of neighbor cells (Fig. 4): (a) *direct*, i.e., neighbors with shared edges; (b) *indirect*, i.e., neighbors with common vertices; and (c) *remote* neighbors. The level of proximity to the cell of reference characterizes remote neighbors.

For instance, level-one (level-two) remote neighbors are the cells, which are direct or indirect neighbors of the direct or indirect neighbors of the cell of reference (of the level-one remote neighbors of the cell of reference). Interconnecting the direct neighbors leads to a set of four directions of movement from each node (*rook's move* is allowed – Fig. 5a). Interconnecting the indirect neighbors adds another set of four directions (*queen's move* is allowed – Fig. 5b). Interconnecting the level-one remote neighbors adds another set of

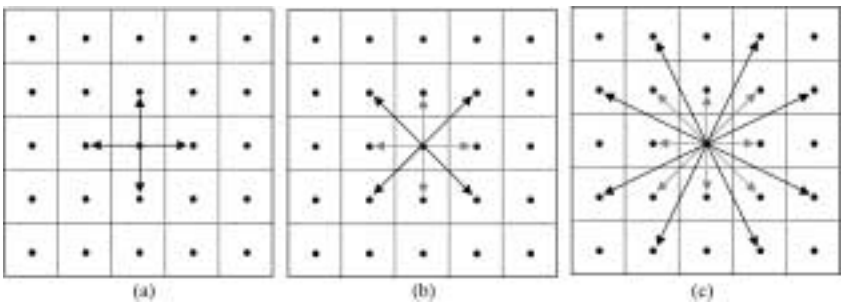

Fig. 5. Four (a), eight (b) and sixteen (c) directions of movement.

eight directions of movement (*queen's+knight's moves* are allowed – Fig. 5c). An exhaustive network would interconnect all direct, indirect and remote (of any level) neighbors.

2.2. *Formulation of the Travel Cost Model*

The *travel cost model* assigns weights to the edges of the network established in the previous step. Its form depends on both the space under study and the application needs. Some representative examples of travel cost models are:

- ✓ the model of *distance* (assign the overall distance),
- ✓ the model of *time* (assign the overall time),
- ✓ the model of *expenses* (assign the overall expenses),
- ✓ the model of *risk* (assign a measure for the overall risk).

In each case the space under study consists of areas that are characterized by a weight, which indicates the cost of movement across them per unit of movement; and depends on the travel cost model in use. A detailed analysis is can be found in (Stefanakis and Kavouras, 2002).

In this paper we consider the cost model of *Euclidean distance* of the two nodes connected by the edge of reference. More sophisticated models can be easily applied.

2.3. *Computation of the Temporal Intervals During which Nodes and Edges are not Accessible*

After the network has been established, the obstacles are considered in order to compute all those temporal intervals during which nodes and edges are not accessible. This information is needed when the network is solved in Step 4, so that the schedule for the moving object is determined in Step 5.

All nodes and edges locations are compared against all moving obstacles locations in time. At the end of this comparison, each individual node and edge of the network is assigned a list of temporal intervals during which it is not accessible, because an obstacle intersects it.

Fig. 6 presents an example of two nodes *A*, *B* and the edge *A_B* connecting them. An obstacle moves from point K to point L. As it is shown, the obstacle covers node *A* during the temporal interval $[t_2, t_3]$ and intersects the edge *A_B* during the temporal interval $[t_2, t_4]$. During these temporal intervals the corresponding node and edge are not accessible.

2.4. *Solving the Network*

After the completion of the previous step, each individual node and edge of the network is assigned a list of those temporal intervals during which it is not accessible. In this step the actual solving of the network is performed. For this reason, appropriate algorithms available in graph theory and artificial intelligence can be applied. In our study we make use of *Ford's algorithm* (Ford and Fulkerson, 1962).

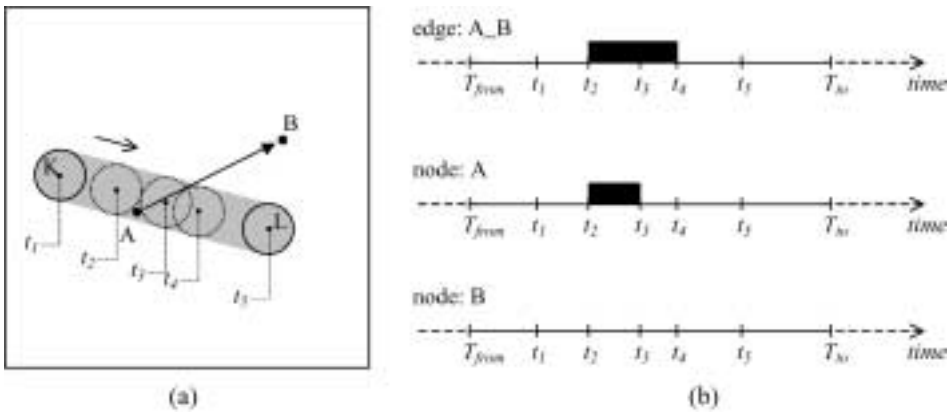

Fig. 6. An example of a moving object (a), and the temporal intervals during which nodes *A,B* and edge *A_B* are not accessible (b).

Provided a graph $G(N, E)$ (where N, E the sets of nodes and edges constituting the graph respectively), and $c_{(m,n)}$ the cost of traversing the edge *m_n*, starting from node m and ending to node n, Ford suggests the following algorithm to find the minimum accumulated cost of each network node n (denoted by $C[n]$) for the trip from a start node n_o:

```
begin
   C[n_o] ←0;
   for each n∈N − {n_o} do C[n] ← ∞;
   while there is an edge (m,n) such that
      C[n] > C[m] + c_(m, n) do
         C[n] ← C[m] + c_(m,n)
end
```

The complexity of Ford's algorithm depends on the number of both the nodes and edges of the network and is equal to $\mathrm{O}(|N||E|)$.

In this study, we extend Ford's algorithm to solve the spatio-temporal network. By executing the algorithm, each node of the network is assigned a list of temporal intervals, during which the node is accessible from the moving object with the minimum accumulated cost for the trip from the start node n_o. We call these intervals as *accessible temporal intervals*. Obviously, the temporal intervals during which nodes and edges are not accessible (computed in the previous step) are taken into account.

Fig. 7 presents the idea through a simplified example. The network (part of) consists of four successive nodes, which are connected through three edges (Fig. 7a). The temporal interval during which the network is considered is [0, 200] (time units). We call this period of time as *network life*. The moving object departs from node 1 (start node) at time 10. The duration and cost of traversing each edge of the network are constant during the network life (Fig. 7b). However, these nodes and edges are not always accessible. Moving obstacles may constraint the traversing during some temporal intervals. These non-accessible temporal intervals have been computed in the previous step (Fig. 7c).

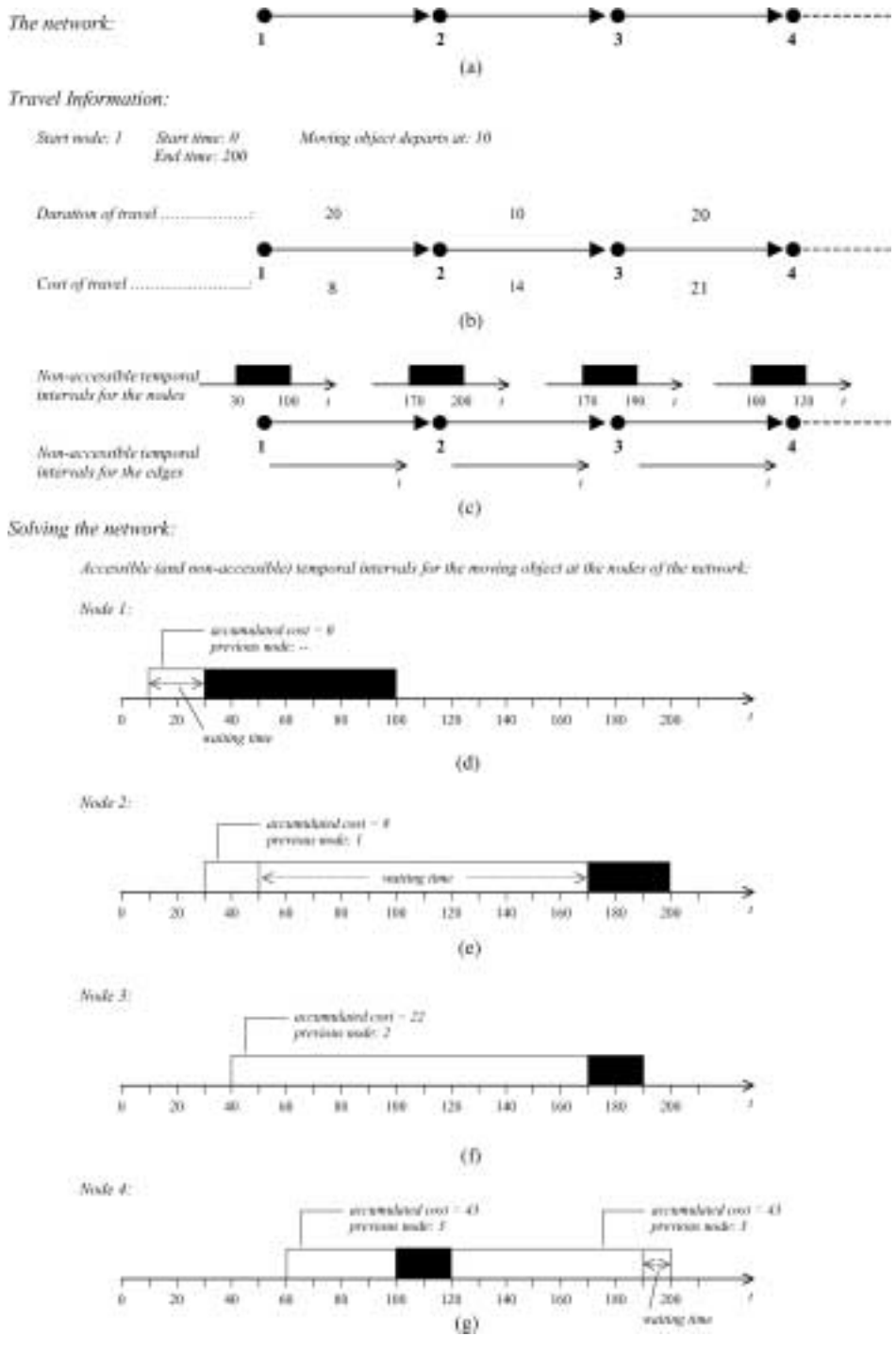

Fig. 7. A simplified scenario.

Provided that the trip starts at time 10, the moving object can reside at node 1 during part or the whole interval defined by time 10 and the next time when instance node 1 is not accessible. Therefore, the accessible temporal interval for the moving object at node 1 is [10, 30] (Fig. 7d,c). The accumulated cost for node 1 is equal to 0 and there is no previous node.

Considering the movement along the edge 1_2, the following apply. The moving object may depart from node 1 at any time during the interval [10, 30]. The edge 1_2 is accessible all the time (Fig. 7c). Provided that the duration of traversing edge 1_2 is equal to 20, node 2 can be reached at any time during the interval [30, 50]. Node 2 is accessible all this period. The accumulated cost at node 2 will be $0 + 8 = 8$. Additionally, the moving object may reside at node 2 until the next time instance when the node is not accessible. Therefore, the accessible temporal interval for the moving object at node 2 is *extended to* [30, 170] (Fig. 7e).

Considering the movement along the edge 2_3, the following apply. The moving object may depart from node 2 at any time during the interval [30, 170]. The edge 2_3 is accessible all the time (Fig. 7c). Provided that the duration of traversing edge 2_3 is equal to 10, node 3 can be reached at any time during the interval [40, 180]. The accumulated cost at node 3 will be $8 + 14 = 22$. However, node 3 is not accessible during the interval [170, 190]. Therefore, the accessible temporal interval for the moving object at node 3 is *reduced to* [40, 170] (Fig. 7f).

Considering the movement along the edge 3_4, the following apply. The moving object may depart from node 3 at any time during the interval [40, 170]. The edge 3_4 is accessible all the time (Fig. 7c). Provided that the duration of traversing the edge 3_4 is equal to 20, node 4 can be reached at any time during the interval [60, 190]. The accumulated cost at node 4 will be $22 + 21 = 43$. However, node 4 is not accessible during the interval [100, 120]. Therefore, the object may depart from node 3 during the intervals [40, 100–20] (or [40, 80]) and [120, 170] (in here we assume that departure from a node is not allowed when the edge to traverse and the opposite edge node are not accessible). This results in the accessible temporal interval for the moving object at node 4 being *split to* [60, 100] and [120, 190] (Fig. 7g). Additionally, the latter is extended to [120, 200], based on the previous discussion.

All accessible temporal intervals and accumulated costs of the nodes at the example in Fig. 7 are subject to change during the iterative execution of Ford's algorithm and the consideration of other nodes and edges in a more complex network. Fig. 8 provides an example of such a change. One node and two edges are added at the network in Fig. 7, i.e., node 5, edge 1_5 and edge 5_3. Node 5 is accessible all the time during the network life; as well as edge 5_3. On the other hand, edge 1_5 is not accessible during the temporal interval [0, 20] (Fig. 8a).

Fig. 8b shows the accessible temporal interval for node 5. The moving object may depart from node 1 at any time during the interval [10, 30]. However, edge 1_5 is not accessible during the interval [0, 20]. Hence, the object must wait at node 1 until $t = 20$. Provided that the duration to traverse the edge is equal to 20, node 5 can be reached at any time during the interval [40, 50]. This interval is extended appropriately (to the end of the network life), since node 5 is always accessible.

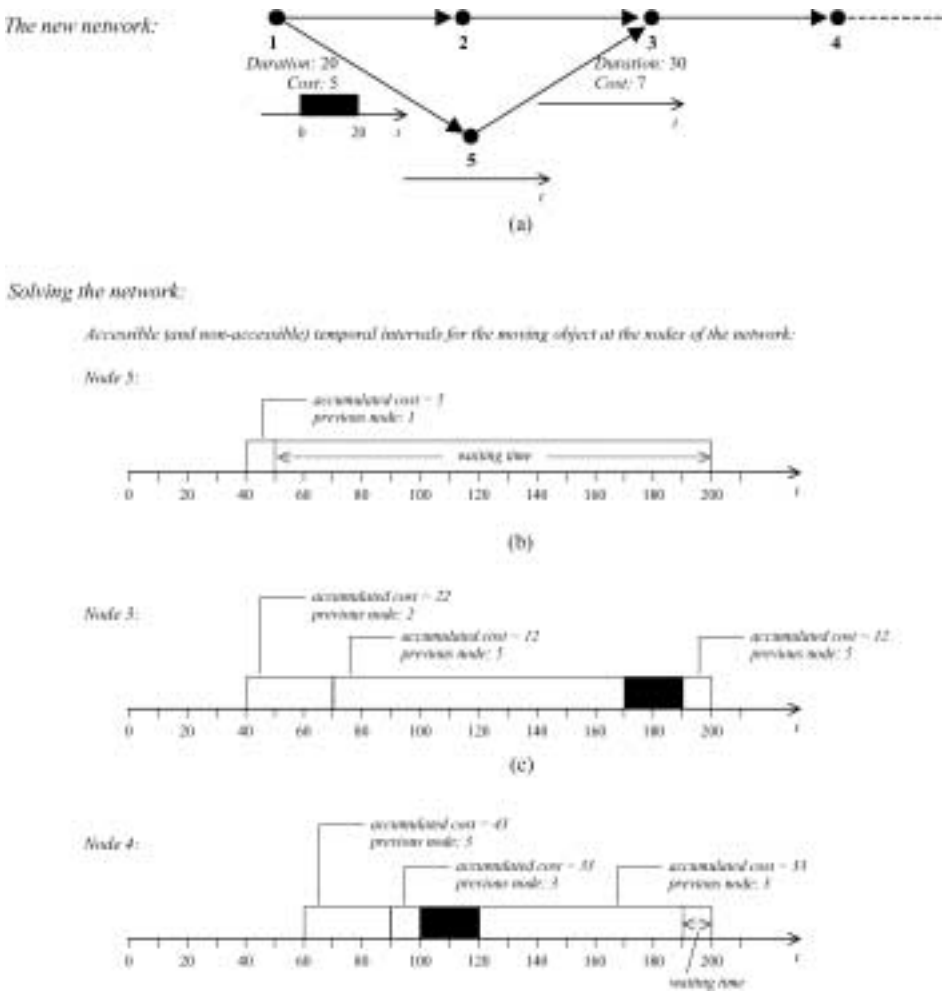

Fig. 8. A more complex network.

Node 2 remains unchanged (Fig. 7e). Node 3 is now accessible from both nodes 2 and 5. New accessible temporal intervals for the moving object at node 3 are computed by considering the edge 5_3. These are intersected to the ones shown in Fig. 7f. The intervals with the minimum accumulated cost dominate. The result of the intersection is shown in Fig. 8c. Notice that, in the new network, node 3 can be accessed from node 2 for the temporal interval [40, 70], with an accumulated cost of 22; and node 3 for the temporal interval [70, 170] and [190, 200] with an accumulated cost of 12.

Node 4 is assigned new accessible temporal intervals according to the new state of node 3. As shown in Fig. 8d, three accessible temporal intervals are assigned to node 4: [60, 90], [90, 100] and [120, 200], with accumulated costs of 43, 33 and 33, respectively.

2.5. *Determination of the Schedule*

After the network has been solved, the schedule for the moving object can be determined, taking into consideration the criteria of movement. Specifically, the previous step has generated for each network node j (where $j = 1, 2, \ldots, N$) a set of n consecutive and disjoint temporal intervals, during which node j is accessible (reachable) by the moving object. Each of these intervals $[t_{ji-from}, t_{ji-to}]$ (where $i = 1, 2, \ldots, n$) has assigned the accumulated cost of movement for the trip from the start node s (c_{ji}) and the corresponding previous node (p_{ji}), i.e.,

$$\text{node } j\text{: } \big\{[t_{j1-from}, t_{j1-to}, c_{j1}, p_{j1}], [t_{j2-from}, t_{j2-to}, c_{j2}, p_{j2}], \\ [t_{j3-from}, t_{j3-to}, c_{j3}, p_{j3}], \ldots, [t_{jn-from}, t_{jn-to}, c_{jn}, p_{jn}]\big\}.$$

For example node 4 at the simplified network of Fig. 8 has been assigned the following set of temporal intervals:

node 4: $\big\{$ [60, 90, 43, 3], [90, 100, 33, 3], [120, 200, 33, 3] $\big\}$.

The set of temporal intervals can be exploited appropriately to determine the schedule. The next paragraphs describe some common scenarios.

In order to schedule the *minimum cost trip*, we choose the temporal interval i at the destination node d with the minimum c_{di} value. Then, we add to the empty stack T – describing the trip – the following triplet:

$$< t_{di-from}, c_{di}, loc_d >,$$

where loc_d is the location of the destination node. The process is applied recursively to the node p_{di}, etc., until start node s is reached. The triplets at T describe the movement of the object. Notice that the cost assigned to the edges of the network may be their length or the expenses to traverse them (e.g., expressed in petrol consumption), etc. In the former case, the shortest in length trip is scheduled. In the latter case, the least expensive trip is scheduled.

At the example of Fig. 8, assuming that node 4 is the destination node, if we look for the minimum cost trip (the lowest accumulated cost at node 4), we choose:

$<$ node 4, accumulated_cost $= 33$, arrival_at 90, previous_node 3 $>$

Then recursively we choose:

$<$ node 3, accumulated_cost $= 12$, arrival_at $90 - 20 = 70$, previous_node 5 $>$,
$<$ node 5, accumulated_cost $= 5$, arrival_at $70-30 = 40$, previous_node 1 $>$,
$<$ node 1, accumulated_cost $= 0$, arrival_at $40-20 = 20$, previous_node n/a $>$.

Therefore the minimum cost path is: $4 - 3 - 5 - 1$, with accumulated cost at node 4 equal to 33, departure from node 1 at $t = 20$ and arrival at node 4 at $t = 70$.

In order to schedule the *fastest trip*, we choose the first temporal interval assigned to the destination node d. Then we add to an empty stack T – describing the trip – the following triplet:

$$< t_{d1-from}, c_{d1}, loc_d > .$$

The process is applied recursively to the node p_{d1}, etc., until start node s is reached. The triplets at T describe the movement of the object.

At the example of Fig. 8, assuming that node 4 is the destination node, if we look for the fastest trip (node 4 is reached the earliest possible), we choose:

$<$ node 4, accumulated_cost $=$ 43, arrival_at 60, previous_node 3 $>$.

Then recursively we choose:

$<$ node 3, accumulated_cost $=$ 22, arrival_at $60 - 20 = 40$, previous_node 2 $>$,
$<$ node 2, accumulated_cost $=$ 8, arrival_at $40 - 10 = 30$, previous_node 1 $>$,
$<$ node 1, accumulated_cost $=$ 0, arrival_at $30 - 20 = 10$, previous_node n/a $>$.

Therefore the minimum cost path is: $4 - 3 - 2 - 1$, with accumulated cost at node 4 equal to 43, departure from node 1 at $t = 10$ and arrival at node 4 at $t = 60$.

In order to schedule the *trip that takes the moving object at destination at time* t_a, we choose the temporal interval k, assigned to the destination node, which contains time t_a (i.e., $t_a \in [t_{dk-from}, t_{dk-to}]$) Then we add to an empty stack T – describing the trip – the following triplet:

$$< t_a, c_{dk}, loc_d > .$$

The process is applied recursively to the node p_{dk}, etc., until start node s is reached. The triplets at T describe the movement of the object.

3. Prototype System and Example Scenarios

A prototype system has been developed in Java to implement the algorithm described in the previous Section. Fig. 9 presents the interface of the system. What is shown is the snapshot of the situation at time 114 (for a network life of [0, 400]). Four moving obstacles are alive then. The current location of the moving object is at point c. Start and destination nodes are marked with s and d respectively.

Fig. 10 shows the values of the parameters that are read by the system for the example in Fig. 9. The spatio-temporal cube is defined by the triplet (0, 0, 0) and (1000, 1000, 400). Cell size is equal to 100×100 (Fig. 3), and 16 directions of movement are considered (Fig. 5c). Start and destination points are located at (0, 400) and (1000, 100) respectively. Their closest network nodes are considered as start and destination nodes (see Fig. 10). The start time for the trip is equal to 0. The velocity of the moving object is constant and equal to 5 distance_units/time_units. The system is asked to compute only the minimum

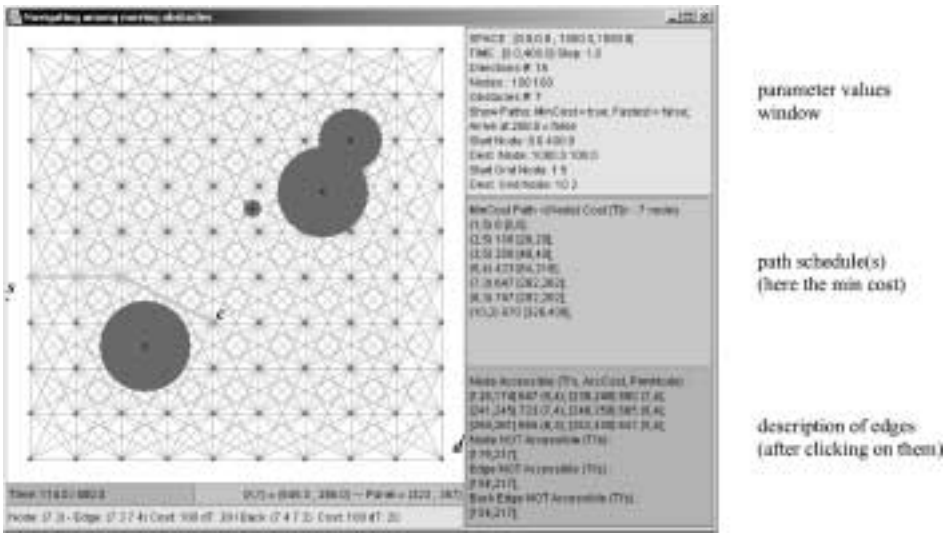

Fig. 9. The interface of the system.

cost schedule. Eight moving obstacles are considered. The first of them (id $= 1$) has a radius of 60 units, and moves from point (0, 300) to point (300, 300) during the temporal interval [0, 60]. The costs assigned to the edges of the network are equal to their length (Euclidean distance).

Fig. 11 presents three schedules for the moving object based on different criteria. With the parameter values listed in Fig. 10, the system is asked to simulate the courses of the moving object which satisfy:

- ✓ Course M: the minimum cost trip (i.e., the minimum length trip; given that our cost model is based on the distance measure).
- ✓ Course F: the fastest cost trip (i.e., the trip that takes the moving object at destination node at the earliest possible).
- ✓ Course A: the arrive at 260 trip (i.e., the trip that takes the moving object at destination node at time 260).

Table 1 presents the courses computed by the system. Each course is described by a set of records:

$$< (x,y)c[t_a, t_d] >,$$

where (x, y) are the coordinates of the nodes composing the course (grid coordinates), c the accumulated cost at the node (for the trip from the start node), t_a the time of arrival at the node, and t_d the time of departure from the node.

```
SPACE <XLL YLL XUR YUR>:
0. 0. 1000. 1000.
TIME <from to granularity>:
0. 400. 1.
NODES <\#inX \#inY>:
100. 100.
DIRECTIONS 4, 8, 16, 32... etc.:
16
START / DESTINATION POINTS <Xstart Ystart Tstart Xdest Ydest>:
0. 400. 0. 1000. 100.
VELOCITY (DISTANCE UNITS / TIME UNITS):
5.
SHOW <MINCOST FASTEST ARRIVE\_AT time> PATHS: (0=false, 1=true)
1 0 0 260.
MOVING OBSTACLES <ID BUFFER \{Xi Yi Ti\}>:
1 60. 0. 300. 0. 300. 300. 60.
2 60. 300. 300. 60. 100. 0. 90.
3 100. 600. 0. 30. 0. 600. 200.
4 50. 500. 100. 300. 800. 200. 400.
5 70. 750. 750. 100. 750. 750. 300.
6 20. 600. 600. 50. 500. 600. 150.
7 80. 800. 0. 150. 800. 800. 250.
8 100. 700. 700. 100. 600. 0. 250.
```

Fig. 10. The list of parameter values.

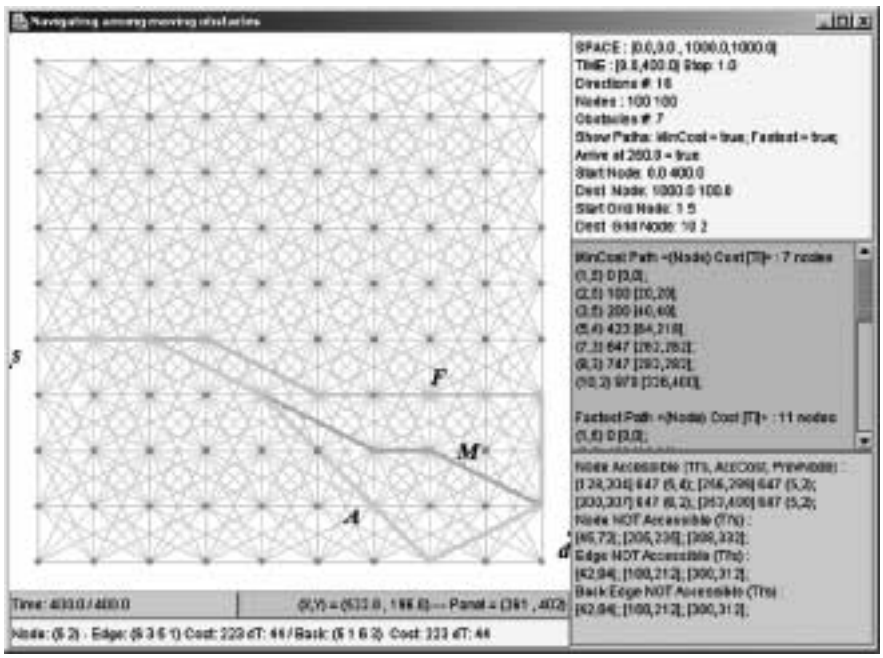

Fig. 11. Three schedules for the moving object based on different criteria.

Table 1
Three courses descriptions for the moving object in Fig. 11

Min Cost Trip	**Fastest Trip**	**Arrive at 260 trip**
(1, 5) 0 [0, 0];	(1, 5) 0 [0, 0];	(1, 5) 0 [0, 0];
(2, 5) 100 [20, 20];	(2, 5) 100 [20, 20];	(2, 5) 100 [20, 20];
(3, 5) 200 [40, 40];	(3, 5) 200 [40, 40];	(3, 5) 200 [40, 40];
(5, 4) 423 [84, 218];	(4, 5) 300 [60, 60];	(5, 4) 423 [84, 84];
(7, 3) 647 [262, 262];	(6, 4) 523 [104, 104];	(6, 3) 565 [112, 112];
(8, 3) 747 [282, 282];	(7, 4) 623 [124, 124];	(7, 2) 706 [140, 165];
(10, 2) 970 [326, 400];	(8, 4) 723 [144, 144];	(8, 1) 847 [193, 216];
	(9, 4) 823 [164, 164];	(10, 2) 1071 [260, 400];
	(10, 4) 923 [184, 184];	
	(10, 3) 1023 [204, 204];	
	(10, 2) 1123 [224, 400];	
Summary:	Summary:	Summary:
Nodes: 7	Nodes: 11	Nodes: 8
Acc. Cost 970	Acc. Cost 1123	Acc. Cost 1071
Arrival at: 326	Arrival at: 224	Arrival at: 260

4. Conclusion

This paper introduces an algorithm for scheduling the trajectory of a point object, which moves on a plane surface comprising of moving obstacles. The schedule may be based on various quantitative criteria, e.g., minimum cost course, fastest course, arrive at specific time course, etc. A prototype system that implements the algorithm has been developed and tested through different scenarios.

Future research directions include: (a) the optimal scheduling of trajectories for a set of moving points (notice that in this case, each moving point constitute an additional obstacle to all other points); (b) the consideration of dynamic travel cost models (subsection 2.2), and (c) the optimization of the algorithm.

Acknowledgments

This work has been done while the author was at the Institut Autonome Intelligente Systeme, Fraunhofer Gesellschaft, Germany, supported by an ERCIM fellowship.

References

Church, R.L., S.R. Loban and K. Lombard (1992). An interface for exploring spatial alternatives for a corridor location problems. *Computers and Geosciences*, **18**, 1095–1105.

Douglas, D.H. (1994a). Least-cost path in GIS using an accumulated cost surface and slopelines. *Cartographica*, **31**(3), 37–51.

Douglas, D.H. (1994b). The parsimonious path based on the implicit geometry in gridded data and on a proper slope line generated from it. In *Proceedings of the 6th International Symposium on Spatial Data Handling*. Edinburgh, Scotland. pp. 1133–1140.
Ford, L.R., and D.R. Fulkerson (1962). *Flows in Networks*. Princeton Univ. Press.
Gibbons, A. (1985). *Algorithmic Graph Theory*. Cambridge University Press.
Goodchild, M.F. (1977). An evaluation of lattice solutions to the problem of corridor location. *Environment and Planning A*, **9**, 727–738.
Johnson, D.B. (1977). Efficient algorithms for shortest paths in sparse networks. *Journal of the Association of Computing Machinery*, **24**, 1–13.
Laurini, R., and D. Thompson (1992). *Fundamentals of Spatial Information Systems*. Academic Press Ltd.
Lindgren, E.S. (1967). Proposed solution for the minimum path problem. *Harvard Papers in Theoretical Geography, Geography and the Properties of Surfaces Series*, **4**.
Mitchell, J.S.B., and C.H. Papadimitriou (1991). The weighted region problem: finding shortest paths through a weighted planar subdivision. *Journal of the Association for Computing Machinery*, **38**, 18–73.
Rich, E., and K. Knight (1991). *Artificial Intelligence*. McGraw-Hill.
Russell, S.J., and P. Norvig (1995). *Artificial Intelligence*: *A Modern Approach*. Prentice Hall.
Sedgewick, R. (1990). *Algorithms*. Addison-Wesley.
Stefanakis, E., and M. Kavouras (1995). On the determination of the optimum path in space. In A. Frank and W. Kuhn (Eds.), *Spatial Information Theory*: *A Theoretical Basis for GIS* (*COSIT 95*). Springer-Verlag. pp. 241–257.
Stefanakis, E., and M. Kavouras (2002). Navigating in space under constraints. *International Journal of Pure and Applied Mathematics* (*IJPAM*), **1**(1), 71–93.
Styliadis, A.D., P.G. Patias and N.C. Zestas (2003). 3-D Computer modeling with intra-component, geometric, quality and topological constraints. *Informatica*, **14**(3), 375–392.
Van Bemmelen, J., W. Quak, M. Van Hekken and P. van Oosterom (1993). Vector vs. raster-based algorithms for cross country movement planning. In *Proceedings of the AutoCarto 11*. Minneapolis. pp. 304–317.
Warntz, W. (1961). Transatlantic flights and pressure patterns. *The Geographical Review*, **51**, 187–212.
Worboys, M.F. (1995). *GIS*: *A Computing Perspective*. Taylor & Francis.

E. Stefanakis holds a dipl. eng. (1992) in surveying engineering from the National Technical University of Athens, Greece; an MScE degree (1994) in geodesy and geomatics engineering from the University of New Brunswick, Canada; and a PhD degree (1997) in electrical and computer engineering from the National Technical University of Athens, Greece. Since 1992, he has been involved in several research projects funded mostly by the E.U. In 2000, he joined the Department of Geography at Harokopio University of Athens, as a professor. He has reviewed many articles for scientific journals and conferences, while he served as a member of organizing, program and scientific committees in international conferences related to geoinformatics. His research interests include geographic information systems, spatial decision support systems, knowledge and database systems, cartography, geovisualization, and Web technology. He has over 30 articles in international journals and conferences in the above areas.

Trajektorijų planavimas plokščiame paviršiuje su judančiomis kliūtimis

Emmanuel STEFANAKIS

Siūlomas taško, judančio plokščiame paviršiuje su judančiomis kliūtimis, trajektorijos planavimo algoritmas. Skirtingi kiekybiniai kriterijai turi būti patenkinti, pavyzdžiui dvi individualias vietas jungiantis kursas turi būti trumpiausias, pigiausias, greičiausias ir panašiai. Yra pristatytas algoritmą realizuojančios sistemos prototipas. Keli pavyzdiniai scenarijai yra aptariami.