AALTO UNIVERSITY
Systems analysis laboratory

Reactive vehicle routing using a shortest path network optimization algorithm

Lasse Johansson

Espoo, September 8th, 2011

Instructor: Ph.D. Kai Virtanen

Aalto-yliopisto

AALTO UNIVERSITY 11000, 00076 Aalto http://www.aalto.fi	ABSTRACT

Author: Lasse Johansson

Title: Optimal flight-path planning using a shortest path network algorithm

Faculty: The Faculty of Information sciences and Technology

Study program: Applied mathematics and Physics

Major: Systems analysis	F3010

Instructor: Kai Virtanen, Ph.D.

Abstract:

In this paper, a label setting shortest path algorithm is used to solve optimal flight paths to a fixed destination for an aircraft flying in an closed rectangular 2-dimensional air space. The air space, which is represented by a symmetric node grid, contains hostile objects that chase the aircraft. The coordinates of both the aircraft and its chasers are updated and the problem is solved repeatedly to produce a simulation of the aircraft's trajectory. For this purpose, a Matlab simulation program was created and its general principles and functioning logic is illustrated. One example simulation is then presented in this paper.

Also, a sophisticated targeting intelligence for the chasers is formulated and used in the simulation. Furthermore, the computational effort of the program with respect to the amount of hostile objects and grid size is presented. Two other methods for solving optimal flight paths are discussed. Finally, several ways of improving the presented method is presented.

Date: 6.2.2011	Language: English	Pages: 18

Keywords: Network optimization problem, shortest path, node grid

Contents

Symbols and notations

$\mathcal{N}$	Set of nodes
N	The number of nodes
$i, i \in \mathcal{N}$	Node i
$\mathcal{A}$	Set of arcs
A	The number of arcs
$(i,j) \in \mathcal{A}$	An arc
x_{ij}	An Arc flow
P	A sequence of arcs $((n,i),(i,j),\dots,(k,l),(l,m))$ such that the end node of an arc is always the next start node in the following arc of the path
$s(i)$	ith node in the shortest path from s to t
y_i	the total flow departing from node i, divergence of the node i
a_{ij}	A scalar arc cost of arc (i,j)
$\mathbf{A}$	Matrix containing all arc costs
$s, s(0)$	Start node of the shortest path, node that contains the aircraft
t	Sink node, destination node for the aircraft
V	Candidate list for the Dijkstra shortest path algorithm
d_i	Label of node i for the Dijkstra shortest path algorithm
(x_i, y_i)	Coordinates of the node i

$(x_{(i,j)}, y_{(i,j)})$	Coordinates of the arc (i,j)
m	Network grid size parameter
$d_{(i,j)\,c_k}$	Euclidean distance between arc (i,j) and chaser c_k
$d_{(i,j)t}$	Euclidean distance between arc (i,j) and node t, additional distance cost
w	Weight parameter for the additional distance cost
a'_{ij}	The sum of the arc cost a_{ij} and the additional distance cost
v_s	Speed of the aircraft
v_c	Speed of a chaser
v	Speed relation v_s/v_c
$dt_{s(i)}$	The minimum difference of arrival times in the node $s(i)$ between the aircraft and its chase

1. Introduction

Network models are used extensively in practice, in a variety of different applications which are based on network problems such as max-flow, traveling salesman, vehicle routing, and multi-commodity flow. Collectively, these network problems constitute the most common class of practical optimization problems (Bertsekas, 1998). Arguably, the shortest path network optimization problem is the simplest of the network problems, although the range of applications is nonetheless extensive, covering problems such as data routing, project management and dynamic programming. Furthermore, the shortest path problem is often needed to be solved as a sub problem with the more complex network optimization tasks. Thus, a host of effective yet simple methods to solve the shortest path problem have been introduced.

In this paper, a shortest path algorithm is used to produce a simulation about an aircraft which selects and repeatedly re-evaluates the safest flight path to its destination, while avoiding hostile moving objects that are chasing the aircraft. In Chapter 2, the basic notations related to network problems are presented. With these notations the label setting solving algorithm for the general shortest path problem is explained. In Chapter 3, the problem of finding an optimal flight route while avoiding moving hostile objects to destination is formulated as a shortest path network problem. Furthermore, two other formulation approaches other than optimization problem are shown. In Chapter 4, a simulation result is presented for the aircraft in which the shortest path problem has been iterated while the positions of the aircraft and the hostile objects have been updated each round. Also, the computational effort of the approach is illustrated. Finally, in Chapter 5, several methods for improving the method and its computational speed are discussed.

2. Network optimization problem and the shortest path algorithm

Based on (Bertsekas, 1998), the notations that are needed to define a shortest path problem are briefly presented in this chapter. Furthermore, the shortest path problem is formulated in mathematical terms and a simple label setting algorithm to solve the problem is presented.

2.1 Network notations

A directed graph $G = (\mathcal{N}, \mathcal{A})$, consists of the set of nodes $(\mathcal{N})$ and of the set of directed arcs between the nodes $(\mathcal{A})$. The number of nodes and arcs are denoted by N and A respectively. An arc from node i to j is denoted by (i, j). If an outgoing arc exists from node i to j then (j, i) is an arc where i is called the start node and j is called the end node. It is possible that both arcs (j, i) and (i, j) exist but no more than one arc may exist between a pair of nodes in the same direction.

A forward path P from node n to m in a directed graph is a sequence of arcs $((n, i), (i, j), \dots, (k, l), (l, m))$ such that the end node of an arc is always the next start node in the following arc of this path. It is also possible to express a forward path simply as a sequence of nodes $((i), (j), \dots, (l), (m))$.

The flow of an arc (i, j) is a scalar which is denoted by x_{ij}. Sometimes convenient to allow negative as well as positive values for a flow, but a negative flow can always be changed to a positive flow if the arc direction is swapped in the graph. In this paper an arc exists (j, i) for every existing arc (i, j) and thus, all flows are set to be nonnegative scalars without any loss of generality. The divergence y_i of the node i is the total flow departing from node i less the total flow arriving at i, which is given by

$$y_i = \sum_{j|(i,j)\in\mathcal{A}} x_{ij} - \sum_{j|(j,i)\in\mathcal{A}} x_{ji}, \forall\, i \in \mathcal{N} \qquad (1)$$

It is said that node i is a source if $y_i > 0$ and that node i is a sink if $y_i < 0$. In this paper the network contains only one source and sink. Note that by adding divergences over all $i \in N$, we obtain $\sum_{i \in N} yi = 0$.

2.2 The shortest path problem

Suppose that each arc (i,j) of the graph is assigned a scalar cost a_{ij} and the cost of a forward path is the sum of the costs of its arcs. Given a start and an end node, the shortest path problem is to find a forward path P between the nodes that has the minimum total cost. In many contexts, arc cost can be viewed as physical distances between the nodes. Based on this analogy, the problem is referred to as the shortest path problem and the total cost is sometimes referred to as the length of the path.

In mathematical terms, the shortest path problem can be formed in the following way: Minimize:

$$\sum_{(i,j)\in\mathcal{A}} a_{ij}x_{ij} \tag{2}$$

subject to

$$y_i = \sum_{j|(i,j)\in\mathcal{A}} x_{ij} - \sum_{j|(j,i)\in\mathcal{A}} x_{ji} = \begin{cases} 1, & i = s \\ -1, & i = t \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

$$x_{ij} \geq 0 \; \forall \; (i,j) \; \in \mathcal{A}$$

Where, s is the start node (source) and t is the end node (sink) referred to as the destination node. It is said that a flow is feasible if the constraining equation (3) holds for every $(i,j) \in A$. It can be shown [1] that if the shortest path problem has at least one feasible flow, then an optimal solution to the problem of Eq.2 exists. Feasible flow is always achieved if we set for any forward path P from s to t that

$$x_{ij} = \begin{cases} 1, & (i,j) \text{ is part of } P \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Reader may verify that equation (3) now holds. If a flow of the form of Eq. 4 is an optimal solution for the problem in Eq. 2, then the corresponding path P is the shortest.

2.3 Label setting methods for solving the shortest path problem

There are many different types of algorithms (Bertsekas, 1998, p. 51) that solve the shortest path problem defined in chapter 2.2. In this paper, a relatively simple label

setting method is used, first published by Dijkstra in 1959. In this algorithm, each node is associated with a scalar label d_i and a candidate list V is formed. In a stepwise procedure, a certain node i is removed from the candidate list for which d_i is the minimum label at each step. The process starts with candidate list containing only the node s and terminates when the candidate list is empty.

The Label Setting Method (Dijkstra)

As presented in(Bertsekas, 1998), the shortest path problem in Eq. 2 can be solved with the following algorithm:

Step 1: Let all arc cost be nonnegative. We set

$$V = \{s\},$$

$$d_s = 0, d_i = \infty, \forall\, i \neq s,$$

where d_i is the label of the node i.

Step 2: A node i is removed from the candidate list such that

$$d_i = \min\{d_j\}, j \in V$$

Step 3: For each arc $(i,j) \in A$, if $d_j > d_i + a_{ij}$, we set $d_j = d_i + a_{ij}$ and node j is added to V if it does not already belong to V.

Step 4: If the candidate list is empty, terminate. Go to step 2 otherwise.

It can be shown that the algorithm always terminates and the final non-infinite labels are equal to the length of the shortest path to the corresponding node. The label setting algorithm described above finds not only the shortest path from node s to t but all the shortest paths from node s to each other node $i \in \mathcal{N}$.

3. Problem formulation

In this chapter, a shortest path algorithm presented in Chapter 2.3 is used to solve a minimum cost flight path for an aircraft to its destination. A bounded airspace for the problem, which contains a number of hostile moving objects, is represented by a two-dimensional symmetric network grid. Arc costs in the grid are associated with the proximity of the hostile objects to the arcs.

The aircraft is set to a starting position (x_s, y_s) and it is set to have a destination at (x_t, y_t). At fixed positions in the $\mathbb{R}^2{}_{++}$ plane lie also a number of hostile objects, which in this paper are referred to as the "chasers". While the chasers are able to move freely in the airspace, the aircraft is bound to move in the grid network from node to the next. The grid is a $m \times m$ symmetric network of m^2 nodes, which is illustrated in the Figure 1. The nodes in the grid are arranged so that node 1 is the left-most node in the bottom row and node m^2 lies in the opposite corner of the grid.

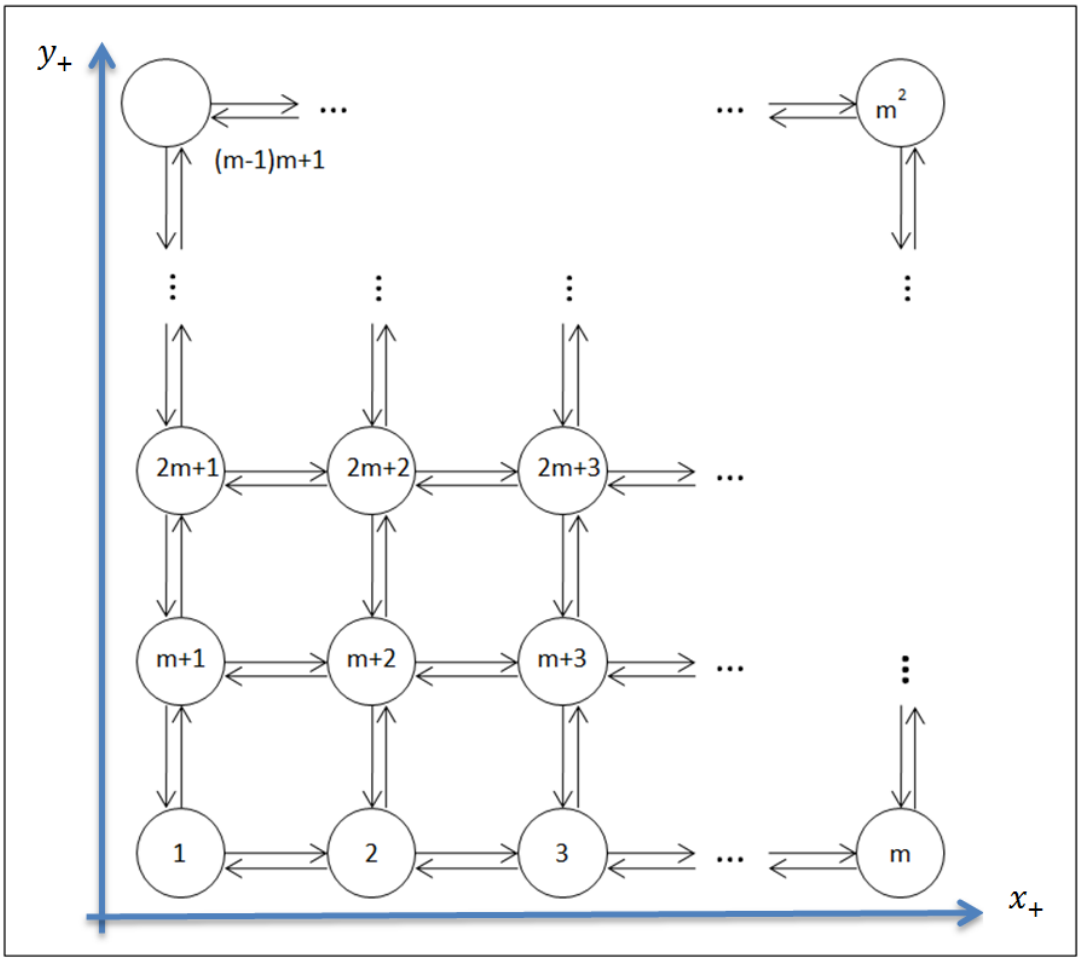

Figure 1: Grid network of m x m nodes. Circles represent nodes in the network and arrows represent the directed arcs between nodes.

For every $(i,j) \in \mathcal{A}$, there are also the arc (j,i) in this grid network. Also as it can be seen from the Figure 1, An inner node i has outward arcs $(i,j-1),(i,j+1),$ $(i,j+m)\ and\ (i,j-m)$ as well as inbound arcs $(j,i-1),(j,i+1),(j,i+m)$ and $(j,i-m)$ - A total of eight arcs.

Each node and arc associated with (x,y) coordinates, which are fixed so that the node 1 lies in the origin as the Figure 1 suggests. The coordinates of an arc are fixed to the center of the arc. Using the grid network's node numbering sequence presented in Figure 1, the x-coordinate of node i are given by

$$x_i = mod(i-1,m) \tag{5}$$

The y-coordinate corresponds to the node's row number less one, which can be calculated with the help of its x-coordinate: The left-most node in the same row of node i is equal to $i - x_i$. Therefore we have that

$$y_i = \frac{i - x_i}{m} - 1 \tag{6}$$

To represent a real air space later on, these coordinates are scaled accordingly. The coordinate y_i should not be mistaken as the divergence of the node i. Indeed, from now on in this paper y_i refers specifically to an y-coordinate of node i and y_{ck} to a chaser k's y-coordinate respectively.

3.1 Arc costs

To emulate the threat from the hostile objects in the form of arc costs, the arc cost should be large near the chasers and smaller far away. A simple way to define the arc costs then would be then to use negative Euclidean distance values between arcs and chasers as arc costs. However, the algorithm presented in Chapter 2.3 requires that each arc cost is to be nonnegative, and thus the arc cost a_{ij} is defined in the following way:

$$a_{ij} = \sum_{k=1}^{n} \frac{1}{d_{(i,j)c_k}} \tag{7}$$

where $d_{(i,j)c_k}$ is the Euclidean norm between the center of the arc (i,j) and the chaser c_k, given by

$$d_{(i,j)c_k} = \sqrt{\left(x_{(i,j)} - x_{c_k}\right)^2 + \left(y_{(i,j)} - y_{c_k}\right)^2} \qquad (8)$$

Eq. 7 and 8 now dictates that if an arc (i,j) is located far away from the chaser's current position, the cost of flowing through the arc is low but near a chaser the arc cost can be substantial because of the nonlinear definition of a_{ij}.

For programming purposes, all arc costs are gathered to a cost matrix **A**, where the cell A_{ij} is the arc cost a_{ij}. If an arc (i,j) doesn't exist, we set $a_{ij} = 0$ making **A** a sparse matrix although a very large one – for a $m \times m$ network grid the cost matrix **A** is of the $size\ m^2 \times m^2$.

It is important to note, that most of the low-total cost paths from s to t have the same number of steps and therefore these paths have the same physical length. Nevertheless, to force the aircraft to favor "shorter" routes, an additional cost is also set for travelling an arc far away from the destination. The adjusted total arc cost a'_{ij} is defined to be the sum of a_{ij} and the Euclidean distance $d_{(i,j)t}$ to the destination node t:

$$a'_{ij} = \sum_{k=1}^{n} \frac{1}{d_{(i,j)c_k}} + w d_{(i,j)t} \qquad (9)$$

where w is a fixed weight coefficient for the additional distance cost. The additional distance cost causes the plane to favor physically shorter routes when the path length is measured on a bigger scale - An example of the phenomenon called as the coastline paradox (Mandelbrot and Benoit, 1983). An example of this effect is presented in Figure 2 in which two paths P_1 and P_2 of equal length from s to t are shown. If the length of these paths are evaluated using only even numbered nodes in the path $s(0), s(2), s(4) \ldots t$, then P_2 clearly has the shortest physical length.

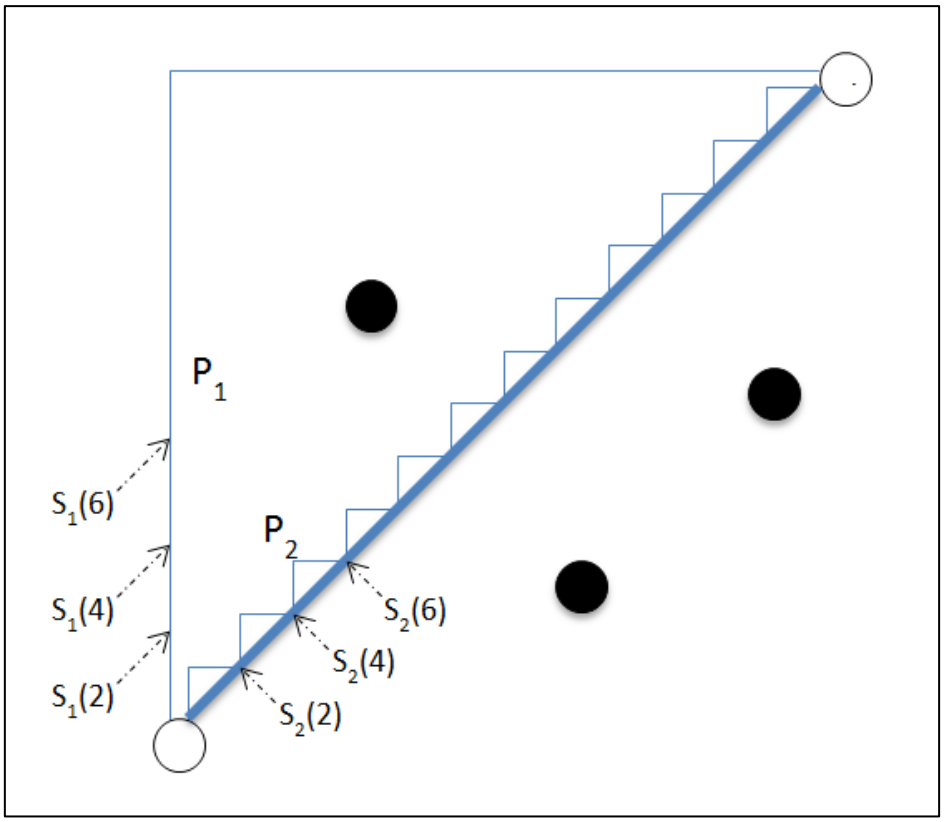

Figure 2: Paths of equal lengths measured with a fractal unit of one arc are not equal when a different fractal unit of two arcs is used.

It can be shown that if the weight parameter of the additional distance cost is set arbitrarily large, then the cost-wise shortest path from s to t is just like the P_2 path in Figure 2, no matter where the chasers are positioned. Because of this, w is set to be relatively small in relation to the chaser costs. However, it should be noted that with even small values of w, the problem of finding the shortest path is transformed essentially to a multiple criteria optimization problem.

3.2 Intelligent targeting system for the chasers

By default, the chasers are set to target the aircraft's next node in the path between the nodes s and t, but also a more sophisticated targeting system for the chasers is presented. In this more advanced targeting mode, the chasers are aware of the aircraft's shortest path and with this information the chasers select which node in aircraft's path would be the most suitable for catching the aircraft and then head in that direction.

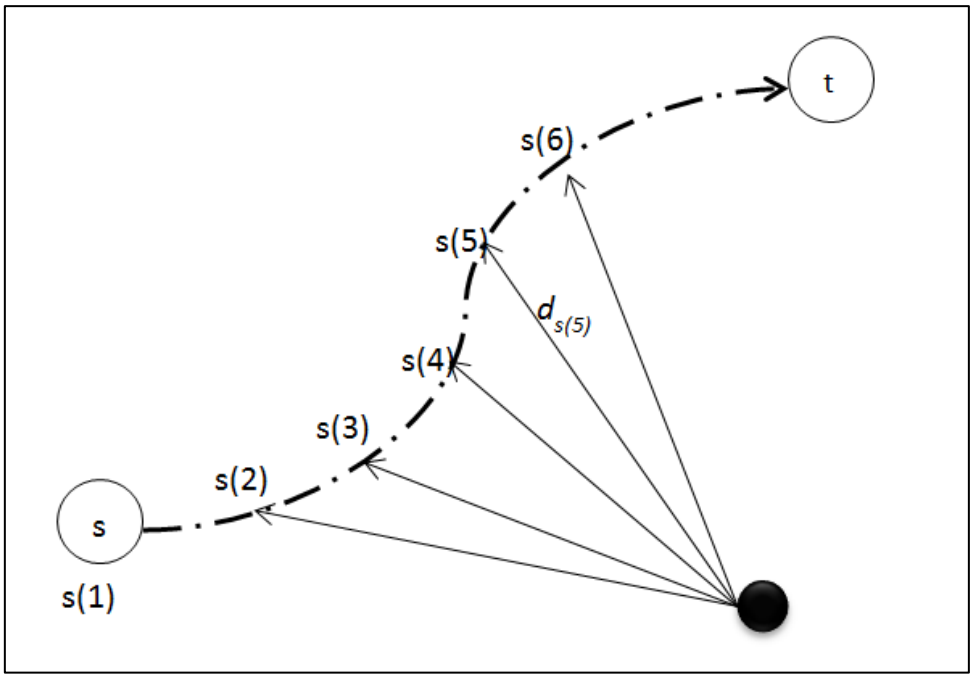

Figure 3: Chaser's intelligent targeting logic which aims to meet the aircraft in some of the nodes that belongs to the aircraft's optimal flight path from $s(1)$ *to* t.

The aircraft's speed v_s in the network is one arc per turn and the speed relation between the aircraft and the chasers is $v = v_s/v_c$. Let $s(2) \dots s(n)$ be the shortest path, illustrated in Figure 3, that the aircraft is about to travel. Then for the aircraft, the time for arriving at node $s(i)$ in the path is simply $i - 1$ turns. If a chaser is located at an Euclidean distance of $d_{s(i)}$ from node $s(i)$, then the chaser is able to arrive at node i in $vd_{s(i)}$ turns. For each node $s(i)$, a time difference $dt_{ck}(i)$ is evaluated, where

$$dt_{s(i)} = \left|(i-1) - vd_{s(i)}\right| \tag{10}$$

If the time difference $dt_{s(i)}$ is, for example, equal to zero, then the aircraft and the chaser are bound to arrive at $s(i)$ simultaneously in $i - 1$ turns if the aircraft doesn't change course at later stages. Therefore, the chasers target such a node $s(i)$ in the path that minimizes Eq. 10.

3.3 Simulation program

A program using Matlab programming software was created to produce simulations of the aircraft's voyage to its destination. The simulation program produces a 3-dimensional plot presenting the steps taken by the aircraft and the chasers and works in the following way:

Step 1: Threat matrix **A** is calculated and converted to a sparse matrix. Both x- and y-axis are scaled so that $m-1$ steps equal a distance of 1000km. Using the start node s and the sparse cost matrix, a shortest paths –problem to all other nodes is solved with algorithm [5]. With the help of this result, the shortest path $(s = s(1), s(2), \dots, t)$ is generated.

Step 2: Aircraft position marked as the node s is updated to be $s(2)$, the next node in the path. The speed of the aircraft is 1000km/h and while the aircraft moves a distance equivalent of 1 arc in a step, the chasers move a distance of $v_c/v_s = 1/v$ arc lengths to the direction defined by their targeting logic. If $s = t$ or step 1 has been taken n times (a fixed maximum number of iterations), the program moves to step 3. Otherwise step 1 and 2 are repeated with the updated positions.

Step 3: Steps taken by the aircraft and the chaser's are presented in a 3 dimensional plot, where z-axis illustrates the passage of time.

3.4 Other possibilities for problem formulation – Dynamic optimization

The shortest path problem formulation used for the simulation is not the only one that could be used. The problem finding the aircraft's path to its destination might as well be presented as a dynamic optimization problem without the network grid. With the methods described in [3], the following analogous problem could be solved:

Minimize

$$J(x(t), y(t)) = \int_0^{t_f} \sum_{k=1} d_{ck}(t) + w d_t(t)\, dt \qquad (11)$$

Subject to

$$f(x'(t), y'(t)) = \sqrt{(x'(t) + y'(t))} - v_s = 0\,, \begin{cases} x(0) = x_s \\ y(0) = y_s \\ x(t_f) = x_t \\ y(t_f) = y_t \end{cases} \qquad (12)$$

where $d_{ck}(t)$ is just like in Eq. 8 except x_s is $x(t)$, y_s is replaced with $y(t)$ and $d_t(t)$ is the Euclidean distance between the destination and the position $(x(t), y(t))$ of the aircraft respectively. The resulting path from Eq. 11 should be smooth in contrast to the

one solved with the discrete network problem optimization method presented in this paper. Also, this alternative method can be easily extended to cover a three-dimensional airspace.

3.5 Other possibilities for problem formulation – MILP

In (Ma, C.S. and Miller, R.H., 2006) another solution technique for optimal path planning using a Mixed-Integer Linear Programming problem formulation (MILP) was presented. The basic obstacle (or threat object in this case) avoidance problem is presented as a linear programming problem where some of the variables are restricted to be integers.

The aircraft's movement is modeled with linear, time invariant discrete equations. Just like in Chapter 3.7, a velocity and destination constraints are presented but also control limits are taken into account in the form of linear equations with slack variables.

The terrain under the airspace is represented with a square based grid pattern but more importantly, the three-dimensional terrain is formed with triangulated irregular networks (TIN) laid on top of the grid pattern. The obstacles with three vertical triangular side walls can be presented in mathematical terms as collision constraints, again with linear equations.

With the methods presented in [4], it would be possible to solve a variation to the shortest path problem by representing the chasers as three dimensional obstacles. With appropriate cost function, the linear optimization problem could be solved with commercially available MILP solver such as CPLEX. There are also several techniques to reduce the computational requirements, such as receding time horizon and multiple time scales – techniques that enable the problem to be solved in smaller parts. However, the number of constraint equations and variables – and therefore the computational effort - is heavily dependent on the number of obstacles which is not the case in this paper's network optimization method.

4. Simulation results

In this chapter, an example of a single shortest path solution by the algorithm is demonstrated with and without the additional distance cost. Then, an example simulation with three chasers is presented.

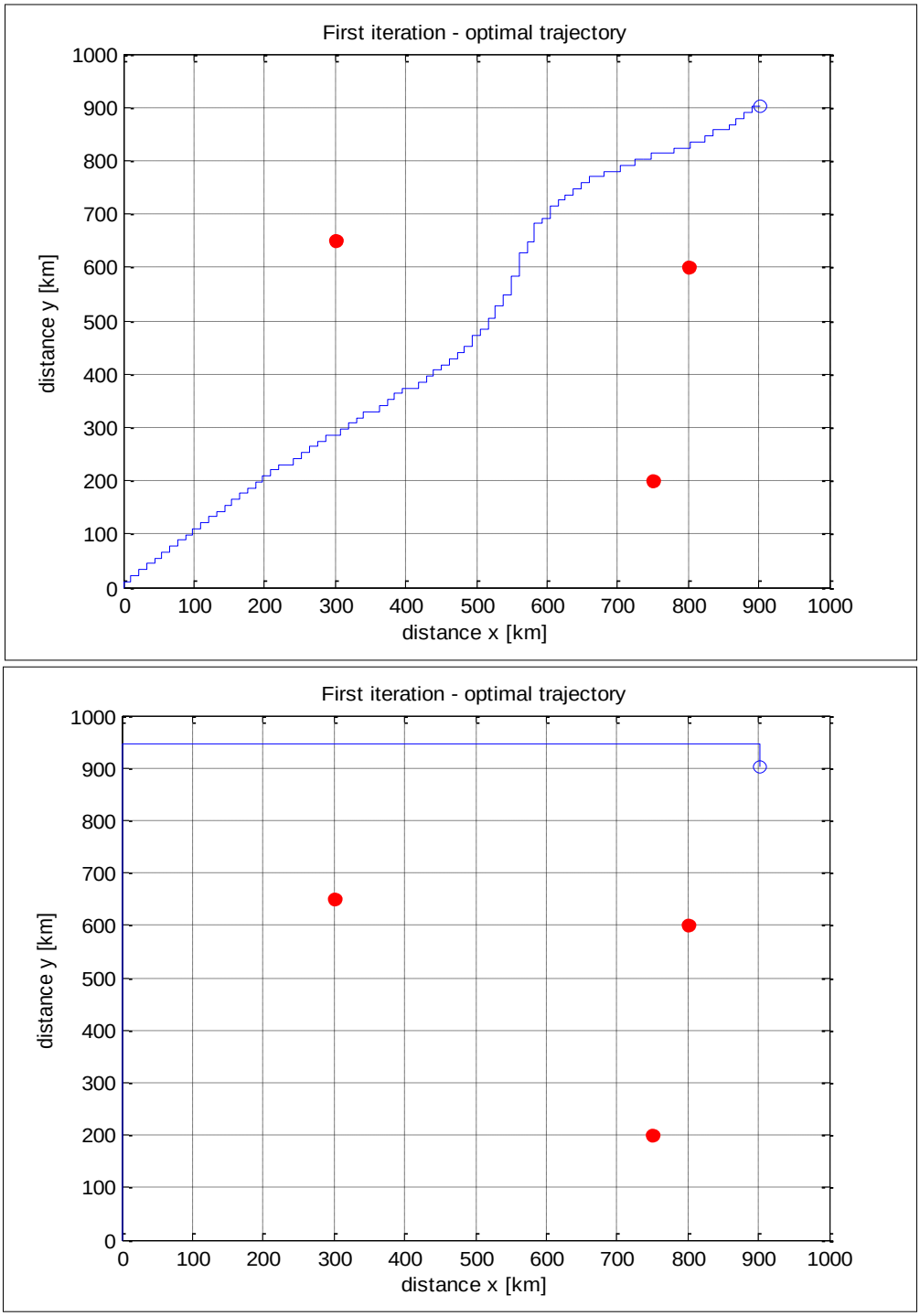

Figure 4a-b: Shortest path to the destination in a 92 x 92 node grid. Chaser positions in the grid are displayed as red dots. In Figure 4a (upper), the additional distance cost weight parameter is set to a value of 0.01 and in Figure 4b, w is set to 0 respectively.

The aircraft has been initially set at (0,0) and has its destination at (900km, 900km) near the upper right corner of the bounded air space. The shortest path to the destination before the first step is shown in Figure 4a with additional distance cost weight parameter w being set to 0.01 and in Figure 4b, the shortest path to the same initial problem is displayed but the weight parameter for the additional distance cost is set to zero. Comparing Figures 5a and 5b shows that the additional distance cost has a significant impact to the shortest path and its smoothed physical length in real air space.

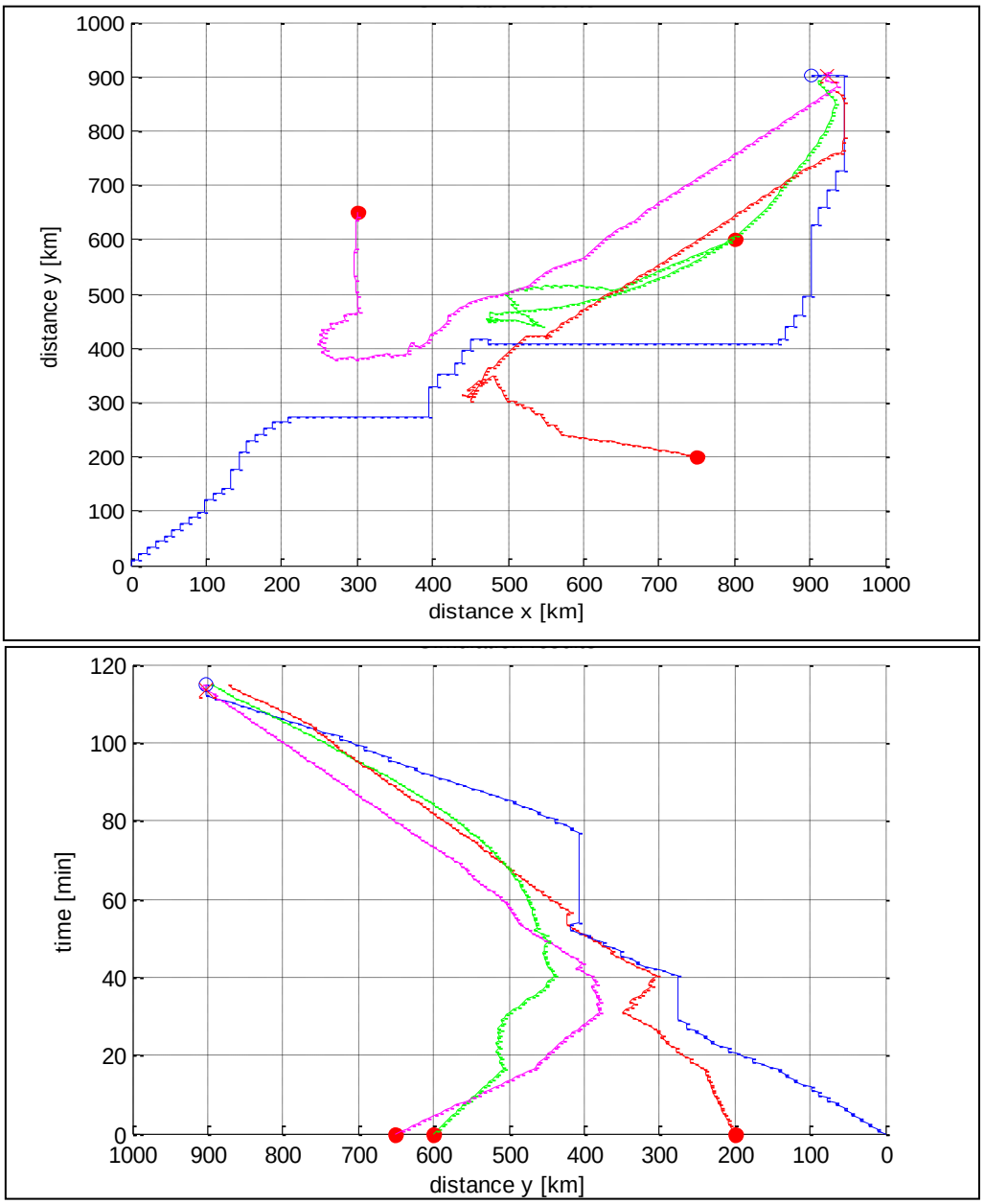

Figure 5a-b: Simulation results using the same parameters as in Figure 5 and setting the number of iterations to 190. Blue curve represents the aircraft's path and other curves represent the three chasers. In Figure 5b (lower), the simulation result is shown from a different perspective to illustrate the passage of time.

With intelligent targeting set on, the result from a full simulation with 190 steps is shown in Figure 5a-b. The aircraft in this case has been set to travel 1.5 times the speed of its chasers. Being a three dimensional plot with z-axis representing the passage of time, the simulation result is displayed from two different angles for visual clarity.

According to the simulation, just a few steps before reaching its destination the aircraft is caught by the intelligent chasers. A small red cross marks the spot where at least one catcher was no more than 0.1 arc's length away from the aircraft at the time. However, according to other simulations with different speed ratios (v_s / v_c) than the 1.5 used in the presented simulation, the aircraft is able to avoid the catchers successfully if the speed relation is increased to 1.8 and up. Remarkably, if the intelligent targeting feature is set off, the chasers cannot catch the aircraft even if the speed relation is set to 1.

4.1 Computational effort considerations

Running the simulation program with various different grid sizes, step counts and number of chasers revealed that the biggest impact on computational effort required is the grid size parameter m. Figure 6 shows how the grid size effects the total computation time for an average modern PC. The simulations were run using 2m steps.

Figure 6: Computation effort using the presented algorithm for simulation. T2 is the total time required for a typical simulation with two chasers consisting 2m steps. T3 is the total time with three chasers.

The curves presented in Figure 6 suggest that the program runs in polynomial time. According to (Bertsekas, 1998), the best estimates of the worst-case running time of the algorithm 2.3 are $O(A + m^2 log\, m^2)$ which is now solved usually n=2m times while the number of arcs **A** is roughly $8m^2$. According to the profiler feature of Matlab the threat matrix calculation and sparsing operation dominates the total computational effort. The number of chasers, on the other hand, does not contribute much to the total effort as they play a part in mostly simple calculations when the arc costs are evaluated with the help of Euclidean norms. Thus it is clearly possible to run simulations using several chasing objects although adding more chasers requires currently the modification of the program source code.

5. Conclusions

With the help of label setting algorithms, a shortest path –problem formulation presented in 2.2, offers a sound method for calculating trajectory to a fixed destination for an aircraft that is set to avoid static objects. Solving the shortest path –problem in a stepwise manner, the method can be used to produce a full simulation of the aircraft's flight while the objects are allowed move in each step.

The simulation program presented in this paper showed that the time required for the computation is heavily dependent on the network grid size. For the test system PC, the maximum grid size parameter bounded by Matlab's memory requirements was 93. Because of this, node separation was more than 10km in the airspace of 1000 square kilometers at best. The reason for the high memory usage is the large cost matrix, which is of the size $m^2 \times m^2$. It is also worth noting that the network grid may not be the best tool for presenting an unbounded airspace. In the example simulation, destination node t was intentionally set to the point (900km, 900km) to allow the aircraft at least some free movement in the vicinity of t. When t was set to the upper right corner on the other hand, the aircraft seemed to push towards that corner in the end no matter where the chasers were positioned.

Even with maximum grid size the aircraft's simulated path was far from being smooth. This was due to the fact that the presented network grid allows only horizontal and vertical stepwise movement. On the bright side, the number of object doesn't contribute much to the computational effort so simulations with even hundreds of chasers are viable.

One of the simplifications of the approach is that the shortest path for a dynamic situation is evaluated using a static threat environment. In many simulations the aircraft was caught by surprise just a couple of steps before reaching the destination because the aircraft's next step in shortest path to t in the vicinity of t included one of the chasers in the next step. Just like a chess player should not plan moves according how pieces are placed in the board but rather how the pieces are likely to be situated in the future in different scenarios, the aircraft should plan ahead to make better path decisions. Of course, with alternative methods presented is 3.7, this problem would still remain.

As it was seen with the demonstration simulation in chapter 3.5, the additional distance cost term and its weight coefficient w has a strong impact to the shortest path. By setting it to value 0.01 the solved shortest path seemed more informative than the solved path while setting w to zero, even though the additional distance cost essentially transforms the initial problem to a multiple criteria optimization problem with arbitrary weightings. This raises a question: what should be the value of w? With appropriate chaser positions the shortest path was actually seen to be quite sensitive to changes in w. At least one thing can be stated about w: the weight of the additional distance cost should be as low as possible so that the initial shortest path is not distorted more than it has to.

5.1 Improvements for the approach

The main disadvantages of the method used for this simulation was the lack of resolution due to the small grid size and also the fixed movement directions available to the aircraft as well as the performance issues with memory. These issues could be dealt with by making the following modifications:

5.1.1 Label setting algorithm improvement

The label correcting algorithm presented in Chapter 2 can be easily modified to a single origin/single destination algorithm by making the algorithm terminate right after the label d_t is removed from candidate list V. With this adjustment it is possible to achieve significant computational savings in solving the shortest path problem if the end node t is relatively far away from s in the graph, which should be the case in many iterations as the aircraft draws nearer to its destination.

5.1.2 Diagonal movement

The network grid could be modified to include diagonal arcs to nearby nodes making the simulated paths appear to be more natural. However, with this modification we arrive at a problem: the time required to travel a diagonal arc is longer than a vertical or horizontal one. Also the distance traveled in a diagonal step would be longer and therefore would make the diagonal movement too low-cost an option for the aircraft to travel than it should. Indeed, simulations with a similar program allowing also diagonal movement showed a path consisting almost nothing but diagonal steps. A simplistic way to deal with this problem could be to weight diagonal arc costs more to counter the greater distance traveled and of course, to enable chasers then to travel farther in a diagonal step.

The grid network presented in this paper presents a two-dimensional airspace, but just as easy as it is to add diagonal arcs, it is possible to construct a three-dimensional grid network by piling up $m \times m$ grid networks on top of each other and adding the appropriate arcs between the layers. With this modification however, the shortest path –algorithm would be dealing with a $m^3 \times m^3$ cost matrix.

5.1.3 Cost matrix size reduction

Because every row of the cost matrix used in the program contains no more than four non-zero cells, it should be possible to increase the maximum grid size significantly and achieve computational effort reductions in the same time by feeding the shortest path algorithm a cost matrix of the form $m^2 \times 4$. If the diagonal arcs are enabled, then the modified cost matrix would be of the size $m^2 \times 8$. This modification requires however, that the general shortest paths -algorithm used in this paper's simulations is to be reprogrammed.

References

[1] Bertsekas, D. 1998. Network Optimization: Continuous and Discrete Models. Athena Scientific. Belmont, Massachusetts. ISBN 1-886529-02-7.

[2] Benoit and Mandelbrot. 1983. The Fractal Geometry of Nature. W.H. Freeman and Co.. pp. 25–33

[3] Kirk, D.E. 2004. Optimal Control Theory, An Introduction. Dover Publications,

[4] Ma, C.S. and Miller, R.H.2006. MILP - Optimal Path Planning for Real-Time Applications. Proceedings of the 2006 American Control Conference. Minneapolis, Minnesota, USA.

[5] Gleich, David. Shortest path algorithm.
http://www.mathworks.com/matlabcentral/fileexchange/authors/23283, Visited 6.2.2011