Real-time replanning of mission routes based upon threats

Master thesis

Division of Automatic Control
Department of Electrical Engineering
Linköping University

Laban Källgren

Reg nr: LiTH-ISY-EX-3126

Supervisors: Christian Johansson Saab Aerospace
            Niklas Ferm Saab Aerospace
            Inger Klein LiTH

Examiner: Inger Klein LiTH

Linköping, 5th of January 2001
Abstract

At Saab Aerospace in Linköping the software and the system of A/C Gripen are developed. One subsystem deals with automatic replanning of mission routes, which in a close future will be integrated with the main system of A/C Gripen.

Modern battlefields are very complex and dynamic, and put extreme pressure on military aircraft attack pilots. In unpredictable situations, the pilot may need to rearrange the predefined mission route, and numerous factors affect the choice of a new route.

In this thesis the possibility of a real-time route replanning support system is examined. Representation of the environment and different quantification methods constitute the base for searching for a new route. A number of different search methods are implemented and simulations of random scenarios are used to evaluate the methods.

According to the simulations A*-search is by far the best method. To make the algorithms faster and save computer memory, A*-search with dynamic operators has also been implemented and evaluated.

Proposals of future work and improvement of the algorithms are treated in this report. Most important is the ability to deal with a dynamic environment, which puts high demands on the reliability of the system.

Sammanfattning


Moderna krigszoner är mycket komplexa och dynamiska, och sätter attackpiloter under hög press. I oföutsägbara situationer, kan piloten tvingas att omplanera den fördefinierade uppdragsrutten. En mängd olika faktorer och parametrar påverkar valet av en ny rutt.

I följande rapport undersöks möjligheterna med ett realtidsbaserat stödsystem för omplanering av uppdragsrutter. Representation av omvärlden och olika diskretiseringsmetoder ligger till grund för att söka efter en ny rutt. En mängd olika sökmetoder implementeras och simuleringar av slumpmässiga scenarion används för att utvärdera metoderna.

Enligt simuleringarna är A*-sökning den metod som ger mest tillfredsställande resultat. För att minska beräkningsstiden och minnesåtgången, har även A*-sökning med dynamiska successoroperatorer implementerats och utvärderats.

Förslag på fortsatt arbete och förbättring av algoritmerna inkluderas i rapporten. Framför allt bör en dynamisk miljö behandlas, vilket ställer höga krav på systemet.
Acknowledgements

This master thesis concludes my studies in Applied Physics and Electrical Engineering at Linköping University. It has been carried out at Saab Aerospace in Linköping between August 2000 and January 2001.

I would like to thank Saab Aerospace for sponsoring this master thesis and the people at Sensor and Countermeasure systems department for enjoyable moments and discussions. Special thanks go to my supervisors at Saab, Christian Johansson and Niklas Ferm, for constructive criticism, interesting ideas and for proofreading the earlier editions of this report.

I would also like to thank my tutor and examiner at the Division of Automatic Control at Linköping University, Inger Klein, for extensive knowledge and support, interesting inquiries and constructive ideas.

I wish to thank Ola Anglesmark at the Department of Computer and Information Science for information about artificial intelligence and search methods. Finally, I wish to thank Hashina Tayyiba who has proofread this report and improved the written language.

Linköping, January 2001

Table of contents

1 Introduction .................................. 1
   1.1 Background ............................... 1
   1.2 Problem description ....................... 3
   1.3 Purpose of the master thesis ............... 4
   1.4 Method .................................. 6
   1.5 Restrictions and simplifications .......... 7
   1.6 Reader’s guide ............................ 7
2 Search strategies ............................ 9
   2.1 Search trees .............................. 10
   2.2 Evaluation function ....................... 11
   2.3 Blind search ................................ 12
      2.3.1 Depth-first strategy .................... 12
      2.3.2 Breadth-first strategy .................. 13
   2.4 Informed search ........................... 15
      2.4.1 Best-first search ....................... 16
      2.4.2 Greedy search .......................... 17
      2.4.3 Uniform-cost search .................... 19
   2.5 A*-search .................................. 20
      2.5.1 Improvement of the evaluation function 21
      2.5.2 Proof of the optimality of A*-search 22
      2.5.3 Proof of the completeness of A*-search 23
      2.5.4 Complexity of A*-search ............... 23
3 The surrounding world ...................... 25
   3.1 Quantification of the geographical map .... 25
   3.2 Representation of threats .................. 26
      3.2.1 Analytic representation of threats .... 27
      3.2.2 HIMM representation of threats ....... 28
   3.3 The tangent graph method .................. 30
      3.3.1 Standard relations of the ellipse ....... 32
      3.3.2 Finding the intersection points ......... 33
      3.3.3 Searching through crucial nodes ......... 35
      3.3.4 Advantages and disadvantages of the tangent 36
1 Introduction

This chapter introduces the reader to realtime route planning for military aircraft, explains the various problems, and defines the purpose of this thesis.

1.1 Background

In a military flight mission, a lot of pressure is put on the pilot. The modern battlefield contains numerous objects and factors which must be taken into account, in order to safely fulfill a mission. Therefore, before a take off, the pilot carefully plans the mission route with regards to all factors. It may be enemy objects such as radar and missile launchers, protection from terrain, weather situation of the mission zone or fuel consumption, that affect the choice of a mission route.

The mission route is presented to the pilot in a tactical (interactive) map as a polygon of straight lines (see figure 1.1). The start of the polygon represents the take off air field, and somewhere on the polygon there is a so called target point where the pilot must arrive at a certain time and with a certain bearing. The time is important for instance when many aircraft are to arrive at the target point at the same moment. Being a few seconds late could mean that the enemy has had time to react which in the end may result in loss of aircraft and harm to the pilot. The bearing guides the pilot to approach the mission target from the correct angle.

In between the start point and the end point of the polygon, there are waypoints which show the pilot when to turn, the speed that is to be kept to reach the goal point at the right moment, the predefined altitude and other information. Most often, the pilot sets the waypoints at landmarks which are easy to recognize, such as buildings and lakes. This makes it easier to remember the waypoints, and the pilot does not have to check the map so often. Another restriction is that the aircraft has physical limits when it comes to sharp turns, maximum speed and increase/decrease of altitude.

The straight path between two waypoints is called a polygon leg. The fact that the pilot can never keep exactly to that leg, makes it essential to include a safety margin or a corridor along the leg. This means that the waypoints must be placed in such a way, that it is possible to follow the route; that is avoiding sharp turns or too short polygon legs, and keeping
Background

Of course it is difficult to optimize the route taking so many different parameters into account, but as the pilot has time before the take off, and always uses computerized aid (a mission support system), a near optimal path can be found which satisfies all restrictions.

As a modern battle field is very dynamic, the situation of the mission scenario may change and make the preplanned route a risky - or even impossible - one. For example, the weather can be very unpredictable, new enemy objects may appear or their position may change due to the uncertainty in the sensor data. In such a case, the pilot needs to rearrange the mission route fast, in order to avoid dangerous zones. Under extreme pressure, the pilot needs to make decisions based on all parameters and restrictions, that in the worst of cases could lead to a severe accident if he fails the replanning. If there is no obvious way to adapt to the new situation, the pilot will probably choose to abort the mission and return to base, instead of getting into an uncontrollable situation.

If the pilot has some support in these kind of situations, he or she could concentrate on more important duties (maintaining correct altitude, manoeuvring radar and so on). The aircraft system itself could calculate and present an alternative route that is risk free and follows all restrictions, or come to the conclusion that the only option is to abort the mission. Such a system would decrease the number of aborted missions and make the flight more secure.

1.2 Problem description

When the situation of the battle field has changed, at first the replanning system must consider if a mission route rearrangement is necessary. This is a matter of decision support agents, and for further reading see [All91] and [Kle98]. If new threats or the weather (or other dynamic objects) do not affect the predefined mission route, a replanning is unnecessary. If the case is different, that is a new threat is observed on the predefined route or a known threat has changed its position, the replanning system is to present a new route which fulfils all requirements of a safe flight during the mission. In figure 1.2 a new threat is detected that affects the predefined route, and the system will search for a new path.

For an effective replanning, the system needs data about the mission zone. Some data comes from sensors, some from the ground control system and some from other aircraft. Other data is the state of the aircraft and the mission goal. It is not obvious which data that is necessary, when and how to use it. It depends on how to replan the route according to the pilot’s desires and the situation. For example, some pilots may wish to follow the predefined waypoints as far as possible while others may be more flexible, depending on their previous experience.

One question of importance is which point (position in map) the route planning is to start from. If the system simply would use the first waypoint of the route, there is a chance that the new route might differ too much from the preplanned one. If the aircraft’s current position is too far away from the new route, it may not be reachable from the current position. On the other hand, if the current position would work as a start point for the route planning, the pilot would most likely need to make complicated manoeuvres (like too sharp turns), to enter the new route. That means that the choice of the start point is important for a smooth change-over between the old and new route.

Problem description

1.2 Problem description

When the situation of the battle field has changed, at first the replanning system must consider if a mission route rearrangement is necessary. This is a matter of decision support agents, and for further reading see [All91] and [Kle98]. If new threats or the weather (or other dynamic objects) do not affect the predefined mission route, a replanning is unnecessary. If the case is different, that is a new threat is observed on the predefined route or a known threat has changed its position, the replanning system is to present a new route which fulfils all requirements of a safe flight during the mission. In figure 1.2 a new threat is detected that affects the predefined route, and the system will search for a new path.

For an effective replanning, the system needs data about the mission zone. Some data comes from sensors, some from the ground control system and some from other aircraft. Other data is the state of the aircraft and the mission goal. It is not obvious which data that is necessary, when and how to use it. It depends on how to replan the route according to the pilot’s desires and the situation. For example, some pilots may wish to follow the predefined waypoints as far as possible while others may be more flexible, depending on their previous experience.

One question of importance is which point (position in map) the route planning is to start from. If the system simply would use the first waypoint of the route, there is a chance that the new route might differ too much from the preplanned one. If the aircraft’s current position is too far away from the new route, it may not be reachable from the current position. On the other hand, if the current position would work as a start point for the route planning, the pilot would most likely need to make complicated manoeuvres (like too sharp turns), to enter the new route. That means that the choice of the start point is important for a smooth change-over between the old and new route.
Another issue is how to deal with the uncertainties of the sensor data. If, for example, the position of a anti aircraft missile site is not a hundred percent accurate, the idea to sneak around that threat would put a risk to the route. There are numerous ways to deal with uncertainties, but as the replanning must be fast enough, a simple way of threat representation is desirable.

During an attack mission, the pilot tries to stick to a low altitude. In this way it is more difficult to discover the aircraft, and enemy missiles have shorter range on low altitude due to the terrain. Therefore the pilot seldom increases the altitude. But of course, there are cases when this might be necessary, for example when the only way to the target point is crossing mountains.

1.3 Purpose of the master thesis

The objectives of this thesis is to explore the possibility of automatic real time route planning based upon threats, and how to deal with different parameters and restrictions. An overview of these factors is given in table 1.1, and probably gives the reader a clue about the complex situation a replanning system has to deal with.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Such as</th>
<th>Pay attention to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enemy objects</td>
<td>Radar, anti aircraft missile site, other aircraft</td>
<td>Uncertainty in sensor data, dynamic range</td>
</tr>
<tr>
<td>Variation of terrain</td>
<td>Hills, mountains, high buildings</td>
<td>Resolution</td>
</tr>
<tr>
<td>Fuel consumption</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Weather situations</td>
<td>Clouds, heavy rain, snow, storm</td>
<td>Fuzzy data, uncertainty in sensor data</td>
</tr>
<tr>
<td>Time restriction</td>
<td>Mission synchronizing</td>
<td>-</td>
</tr>
<tr>
<td>Altitude</td>
<td>Attack level</td>
<td>Threat range dependent of altitude of aircraft</td>
</tr>
<tr>
<td>Approach bearing</td>
<td>Bearing at target point</td>
<td>-</td>
</tr>
<tr>
<td>Logic waypoints</td>
<td>Lakes, hills, buildings</td>
<td>Easily memorized</td>
</tr>
<tr>
<td>Physical limits</td>
<td>Turns, speed, acceleration, length of polygon legs</td>
<td>-</td>
</tr>
<tr>
<td>Safety margins</td>
<td>-</td>
<td>Avoid risk area</td>
</tr>
</tbody>
</table>

Table 1.1 Factors that affect the choice of a mission route.

Another important point of view, is that a support system is worthless if it does not come to the same conclusions as the pilot would under similar circumstances, provided he had enough time to make a rational decision. On the other hand, the system shall also prevent the pilot from making fatal decisions because of stress. Therefore, great effort is put into the investigation of methods of making the system as intelligent as possible, in such a way that it brings forth results that the pilot is contented with.

As the system is to run in a real time environment, there are time restrictions as well. If it took ten minutes to rearrange the route, the help would come too late [Dur97]. To make the calculations fast enough, simplifications must be done, and the question of how to speed up algorithms without lowering the performance to unacceptable levels, is treated.

The summary of this chapter is that with regards to given input parameters such as threats, weather, terrain and aircraft state, the replanning system is to present a new safe route to the target point. The
new route shall not differ too much from a route that the pilot would have planned under the same circumstances, and it must fulfil the physical requirements such as fuel and turns (see figure 1.3).

![Figure 1.3](image.png)

*Figure 1.3* A new mission route is presented that fulfils all requirements such as smooth turns, and the new threat is avoided (compare with figure 1.2).

### 1.4 Method

Automatic route planning is a discipline of Artificial Intelligence. There exists well known algorithms for solving route planning problems, and one which is widely used is A*-search. In an earlier master thesis at Saab [Lin09], this algorithm is proposed as a tool for solving mission replanning based upon threats. This is a very general application though, and special treatments and algorithms must be included to take all requirements for military flight mission replanning into consideration.

Alternative algorithms to A*-search are examined and evaluated. Those will serve as a comparison measure to show the advantages of A*-search.

In order to test and evaluate the result of A*-search and the other algorithms, a simple simulation environment is implemented in C++. The data from the simulations is converted and presented graphically in MATLAB.

An interview has been made with persons that have experience in mission route replanning. This interview serves as a support for evaluating the performance of the replanning system. It also give relevant information on how to adjust the parameters in the optimization process.

A rich variety of books, theses and articles which treats the area of artificial intelligence, route planning, agent support systems and other fields of importance for this thesis have been studied.

### 1.5 Restrictions and simplifications

As mentioned before, the complexity of the system grows rapidly with the many different parameters that affect the choice of a new route. Therefore it is necessary to simplify the simulations in order to extract the relevant issues of mission replanning.

Only two types of obstacles are treated: stationary threats on the ground and elevation of terrain. Weather may also affect a route and is mentioned in the report when suitable, but is not treated in the simulations though.

Even though flying over a threat on a high altitude may be a choice, it is not treated in this report. Threats are regarded as 2-dimensional risk areas.

As the replanning system is a module in a larger system, it is supposed to get information from other modules. In the simulations of this thesis, the information is simply entered as default parameters.

### 1.6 Reader’s guide

*Chapter one* introduces the reader to the purpose of this master thesis. A description of the route planning problem is given, and the method used to solve the problem is presented.

*Chapter two* gives an overview of search strategies. Starting with simple ones, the aim is to improve the strategies in order to find more intelligent routes. The advantages of A*-search is presented.

*Chapter three* deals with the representation of threats and terrain in route planning problems. Storing information in a map grid data base (homogeneous quantification), or finding crucial nodes (heterogeneous quantification) are two approaches.

*Chapter four* gives an explanation of how to improve the results of A*-search with the help of node reduction refinement or by using dynamic operators.
In Chapter five iterative improvement algorithms are presented, which is an alternative strategy to A*-search in order to find routes with a low number of waypoints.

In Chapter six a summary of an interview is presented. This will serve as a basis to refine and adjust the search algorithms in order to find desirable mission routes.

In Chapter seven the reader will find the results of the simulations.

Chapter eight gives the conclusions of the simulations, and an overview of the problems that have been solved.

In Chapter nine the reader will find ideas and suggestions of future work.

2 Search strategies

A*-search (pronounced as A-star-search), is a well-known search algorithm that is, among other things, widely used to solve route planning problems. For example, the next generation of NASA’s planetary rovers use a specialized version of the A*-search to find fast routes in unexplored terrain [Sno99]. In this chapter the reader will get the essence of the A*-search strategy, which is a method that finds an optimal solution of a problem. When nothing else is mentioned, the underlying theory of this chapter comes from [Rus95].

Starting with a simple blind search strategy, the aim of this chapter is to make the search smarter step by step, and to finally end with A*-search. The following problem will serve as a running example through the chapter, and show how different strategies reach various conclusions, and why A*-search is preferable:

Example 1. Plan a route for travelling by car from Karlstad to Linköping. The fastest route is preferable as the driver wants to save time.

Figure 2.1 A simplified map of middle Sweden.

The map in figure 2.1 is a graphical representation of a simple map data base, which contains information about a number of cities in middle Sweden.
Sweden, and the connecting roads between them. The search algorithms use the map data base as an information source in order to find a route, and of course the result of the search is dependent on the goodness and the reliability of the data base. More cities (higher resolution) may improve the result, and later on in this chapter more information is added to the data base, which makes the search more intelligent.

2.1 Search trees

Common for all search methods is that an abstract data type is built up during the search, a so called search tree [Ska96]. There are some technical terms that are widely used in the area of search strategies, and in this section we give a brief overview of these terms. The notations are found in [Caw98] and [Bro98].

A search tree has nodes, which represent different states. In figure 2.2 the nodes are represented by circles, and the states are denoted by letters. In the route finding example in this chapter, a state is simply a city that the driver can visit.

For each node, there are operators connected to other nodes in the tree. An operator is an operation that changes a state into another state, normally represented by lines in the search tree. An evaluation function (see section 2.2) is connected to each operator, and it puts a cost to the state changing action. An operator in example 1, means changing the car position from one city to another.

A search tree has nodes, which represent different states. In figure 2.2 the nodes are represented by circles, and the states are denoted by letters. In the route finding example in this chapter, a state is simply a city that the driver can visit.

For each node, there are operators connected to other nodes in the tree. An operator is an operation that changes a state into another state, normally represented by lines in the search tree. An evaluation function (see section 2.2) is connected to each operator, and it puts a cost to the state changing action. An operator in example 1, means changing the car position from one city to another.

Figure 2.2 A simple search tree.

The children of a node, are all possible states which can be reached from the current state. In figure 2.2, B and C are children of A, and F and G children of C. The parent of a node in its turn, means the opposite direction. For example, B is the parent of D and E.

2.2 Evaluation function

The series of operators to get from a state to another, is called a path. A-B-E is one path and C-F another. A leaf is a node without any children, like D, E, F and G.

Finally, the purpose of a search tree, is to find a solution path from the initial state - the root - to the goal state. In example 1, that is to find a route between Karlstad and Linköping.
2.3 Blind search

Blind search is a category of search strategies, that does not use any evaluation function to decide which node to expand. That means that worse nodes (bad states) may be expanded before the better ones, giving results that often are not optimal. The advantage is that if any route is to be found, i.e. all routes are equally good, blind search strategies are fast. There exist many different variants of blind search, and the two most basic ones are explained here.

2.3.1 Depth-first strategy

This is a strategy that systematically searches through a set of states, and chooses the goal path that is first found. Starting in the root node (initial state), Depth-first keeps going down a path in the search tree until it reaches a leaf or finds the goal state. In case of a leaf, it backtracks in the so-far-path, until it finds another path to track down.

Solving the problem in example 1 - finding a route between Karlstad and Linköping - with the Depth-first strategy, gives the following steps (in figure 2.3, the expansion of the search tree for each step is shown):

1. Standing in the initial state (the root), which is being in Karlstad, find the connecting cities according to the map database and add them as children in the search tree.
2. Here the northmost city is added first (a predefined rule), and the other ones counter clock wise.
3. Move to the leftmost child (Falun) and check its connecting cities. Add these as children to the current node.
4. Move to leftmost child (Gävle). According to the map database, there are no connecting cities to Gävle (except Falun that is the parent node), thus it is a leaf. Backtrack from the leaf to its parent (Falun). Choose next child in order, which is Stockholm, and move there.
5. Expand the node, and add its children to the search tree. Move to the leftmost child (Norrköping).
6. Once again, add children to search tree and move to the leftmost one. Now, we have reached the goal Linköping and the solution path of the problem is presented: Karlstad - Falun - Stockholm - Norrköping - Linköping.

From the final search tree plot in figure 2.3, it is easy to understand why this strategy has the name Depth-first: it goes from the root, deeper and deeper down the tree, until it finds the goal node.

In order to evaluate the result, the total travel time can be calculated. With a constant average speed of 80 km/h the time cost is 
\[
\frac{(231+224+164+43)}{80} = 8 \text{ hours and 15 minutes.}
\]
Anyone familiar with the geography of Sweden, would say that this is a rather bad route. There are definitely better ways between Karlstad and Linköping, but as this is a blind search, one should not hope for better solutions.

2.3.2 Breadth-first strategy

Instead of following a path to a leaf or a goal state (as in Depth-first search), the idea of Breadth-first is that all children of a node are examined before expanding any new ones. This is also a systematic way of searching a tree, but differs from Depth-first search in the way that it requires more memory. Even so, Depth-first yields the risk of not finding a solution, while Breadth-first always does so.
Once again, applying this strategy to example 1, gives the steps (see figure 2.4):

1. Find all connecting cities to Karlstad (the initial state). Add them as children to the root. Use the same order of adding the state as in Depth-first, that is northmost cities first.
2. Check all the nodes from left to right. If none is the goal state (Linköping), find connecting cities to all nodes and add them as children.
3. Once again, check all nodes from left to right, to see if goal state is found. As it is not, the nodes are expanded and their children added to the tree. As node ‘Gävle’ is a leaf, this is unexpandable.
4. Checking the new row of nodes, the Breadth-first algorithm finds the goal state ‘Linköping’ in the rightmost node. The search is done and the solution path presented: Karlstad - Trollhättan - Jönköping - Linköping.

The travel time of this route will be \((151+148+129)/80 = 5\) hours and 20 minutes, which is a great improvement from the route presented in Depth-first. However, this is still not the best route. By taking a glance at the simple map in figure 2.1, one can see that there are faster ways.

Obviously there is a demand for an increased level of intelligence of the search, if a faster trip is to be found. Adding information to the map data base, and using an evaluation function in order to decide which nodes to expand, will significantly improve the result as shown in section 2.4.

2.4 Informed search

In informed search, the decision of the next node to expand, is taken according to the value of the evaluation function. Remember that in blind search, the nodes are expanded in some kind of “top-down” or “left-right” order which does not care how good a state or an operator is. The benefits of informed search, is that the best node in the search tree is expanded first, due to the value of the evaluation function.

The name informed search comes from the external information used to expand nodes in the search tree, to find a better solution path. Not only the operators of the states are available but also their quality.
2.4.1 Best-first search

Best-first search always minimizes the next step of the search. That is, the operator of a node with the lowest cost value, is applied first when adding children to the search tree. In Best-first search, the general evaluation function is defined as follows:

\[ f(n) = \text{path cost between two nodes} \]

or for the example problem

\[ f(n) = \text{road distance between two cities} \]

In example 1, the consequence of this is when standing in one city, the closest neighbour city is chosen as the next destination. Taking a look at figure 2.5 and applying the Best-first strategy, the search gives the result (see figure 2.6):

1. The search starts in the root node, expands it and adds the children.
2. The evaluation function gives that the operator ‘move to state Kristinehamn’ gives the lowest cost.
3. Kristinehamn is expanded, and its nodes added to the tree. Going to Degerfors, gives the lowest cost and therefore it is the next node to expand.
4. Being in Degerfors, the natural choice would be to go to Askersund. Örebro is closer though, so that one will be the next node to explore.
5. In state Örebro, the closest city is Kristinehamn, and should be the next state. However, as it is already visited, one can decide to block it for further investigation. In that way, the fatal result of driving around in circles between Kristinehamn, Degerfors and Örebro is avoided. I.e. the next state to explore is Norrköping, which is closest to Örebro (apart from Kristinehamn).
6. The final step is going to Linköping, that is closest to Norrköping, and the search is done.

The path Karlstad - Kristinehamn - Degerfors - Örebro - Norrköping - Linköping takes 3 hours and 30 minutes to travel, which is a significant improvement from before.

Still, the choice in step 4, of going from Degerfors to Örebro instead of Askersund, seems to be a bad idea. Being in Degerfors, the better choice is to head for Linköping, and not in the opposite direction that increases the distance to the goal state. This is the price to pay, when trying out the closest cities first. It tends to move to local minimas first, before finding the global one. Luckily, with additional information this problem can be avoided.

2.4.2 Greedy search

Instead of moving to the closest city around, as in Best-first search, one can minimize the estimated cost to reach the goal. That is, the node whose state is judged to be closest to the goal state is expanded first. The function that estimates the total path cost of going from a state to the goal state, is called a heuristic function. In Greedy search, the general evaluation function is defined as follows:

\[ h(n) = \text{estimated path cost of going from node n to the goal state} \]

In the case of route planning problems, where the aim is to find a path between two points, a logic heuristic function is the straight-line-
distance between the current state and the goal.

\[ h(n) = \text{straight line distance between city } n \text{ and Linköping} \]

Before trying out this strategy to find a route between Karlstad and Linköping, the information about the straight-line-distances between Linköping (the goal state) and other cities are added to the map data base (see table 2.1).

<table>
<thead>
<tr>
<th>City</th>
<th>Straight-line-distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Askersund</td>
<td>76 km</td>
</tr>
<tr>
<td>Degerfors</td>
<td>135 km</td>
</tr>
<tr>
<td>Falun</td>
<td>280 km</td>
</tr>
<tr>
<td>Karlstad</td>
<td>184 km</td>
</tr>
<tr>
<td>Kristinehamn</td>
<td>152 km</td>
</tr>
<tr>
<td>Mariestad</td>
<td>124 km</td>
</tr>
<tr>
<td>Motala</td>
<td>40 km</td>
</tr>
<tr>
<td>Norrköping</td>
<td>44 km</td>
</tr>
<tr>
<td>Stockholm</td>
<td>196 km</td>
</tr>
<tr>
<td>Trollhättan</td>
<td>224 km</td>
</tr>
<tr>
<td>Örebro</td>
<td>137 km</td>
</tr>
</tbody>
</table>

Table 2.1  Straight-line-distances between Linköping and other cities.

Solving example 1 with this strategy, yields the following steps (see figure 2.7):

1. Standing in Karlstad the agent driver has Falun, Kristinehamn and Trollhättan as connecting cities.
2. Out of these three, Kristinehamn is the one that is closest to Linköping (see table 2.1), and therefore selected as the next state.
3. From Kristinehamn, the driver heads for Mariestad as this is the path that minimizes the estimated cost of going to Linköping.
4. From Mariestad there is only one choice, which is going to Jönköping.
5. In the final step the agent chooses between going to Trollhättan, Motala or Linköping. That choice is obvious, as going to Linköping is the goal of the search (the heuristic function has value zero in the goal state).

The proposed path in this case is in: Karlstad - Kristinehamn - Mariestad - Jönköping - Linköping, with the total travel time 4 hours and 45 minutes. It is not better than the Best-first search, but still faster than the blind search strategies.

2.4.3 Uniform-cost search

The third common informed search method is called Uniform-cost search. Its evaluation function is defined as:

\[ g(n) = \text{total path cost between the root and node } n \]

or for example 1:

\[ g(n) = \text{road distance between Karlstad and city } n \]

This method is optimal and complete, but can be very inefficient. It is very similar to the Breadth-first method, except that it finds the goal state faster and is optimal, as it is provided with external information.

No graphical example is shown in this report, as almost all nodes in the search tree are expanded. When the evaluation function becomes this complex computerized aid is necessary, even though there are only 15 possible states. In the simulations of this thesis, there will be 225 000 possible states!
2.5 A*-search

Uniform-cost search minimizes the cost-so-far value, when choosing which node to expand. Greedy search, on the other hand, minimizes the estimated cost to reach the goal. Unfortunately Uniform-cost search is very slow and inefficient, and Greedy search is not optimal. By combining the ideas of Uniform-cost search and Greedy search, a new evaluation function is developed. By adding the evaluation functions of Greedy search and Uniform-cost search, we get:

\[ f(n) = g(n) + h(n) \]

where \( g(n) \) is the cost of going from the initial state to state \( n \), and \( h(n) \) is the estimated cost of going from state \( n \) to the goal state. The definition is:

\[ f(n) = g(n) + h(n) = \text{total path cost from the root to node } n, \text{ plus estimated path cost between node } n \text{ and the goal state}. \]

or for example 1

\[ f(n) = g(n) + h(n) = \text{road distance between Karlstad and city } n, \text{ plus straight-line-distance between city } n \text{ and Linköping} \]

The heuristic function \( h(n) \), that estimates the remaining path cost, is very optimistic when using the straight line distance between cities. Most likely the road distance is longer, because of curvy roads. However, the optimistic estimation is a requirement if A*-search is to be complete and optimal. In section 2.5.2, the reason of this requirement becomes clear when proving the optimality of A*-search.

Solving example 1 with A*-search gives the following result (see figure 2.8):

1. At first the evaluation function has value 0+184 km. The zero comes from that no distance is travelled yet, and 184 is the straight-line-distance to Linköping (see table 2.1).
2. By expanding the node, the three children of Karlstad are added to the search tree. Here Kristinehamn gets the lowest cost, as it minimizes the travelled path plus the estimated cost of reaching the goal.
3. The children of Kristinehamn are added, and out of these three node Degerfors has lowest cost.
4. Degerfors has two children according to the map data base, and they are added to their parent in the search tree. Going to Askersund has the lowest cost, and therefore this is the next node to expand.
5. Askersund has only one child, which is Motala. From Motala the goal state will be reached, but first the node Mariestad is expanded, because it has a lower evaluation cost than node Motala. This is not plotted on the graph though (only as dashed lines), in order to save space.

After Mariestad is expanded, Motala is the next in turn and the goal node Linköping found. In this case, the travel time is 2 hours and 45 minutes, which actually is the fastest route (according to the author’s experience).

![Figure 2.8 The steps of A*-search. Nodes are labelled with \( f(n) = g(n) + h(n) \).](image-url)

2.5.1 Improvement of the evaluation function

As mentioned before, the evaluation function decides what is an optimal route. In A*-search the evaluation function consists of two parts, the deterministic function \( g(n) \) and the heuristic function \( h(n) \).

The choice of \( h(n) \) in the application of route finding problems is simple. The straight-line-distance is a natural choice, as it estimates the remaining path in an optimistic way.

There are cases though, when more information is needed to improve the search. The secondary requirement of the problem of finding a route between Karlstad and Linköping, was that it should be time cheap. So far the cost function \( g(n) \) is based on the distances between cities, and to
achieve the travel time, the route length is divided with the average speed 80 km/h.

What if there was a local snow storm between Motala and Linköping, which created slippery roads? Most likely, the average speed would be much lower there; maybe so much that it would be a better idea to go to Jönköping first and from there to Linköping. By adding information about speed limits, weather situations, road work, traffic jams etc., the search engine would get smarter, and perhaps present a different route from the one above.

Finally, as mentioned in section 2.2, fuzzy requirements like “I don’t like Örebro” or “I prefer a road next to a lake, instead of highways” are very difficult to deal with. These parameters refer to feelings, unique of the driver, rather than measurable units. One can say, that in order to find a good evaluation function, a physical unit should be used as a measurement, for example time cost.

### 2.5.2 Proof of the optimality of A*-search

The optimality of the A*, means that if there exist many solution paths to the problem, the best one is found first.

Let $G$ be an optimal goal state with path cost $f_o$, and $G_2$ a suboptimal goal state with path cost $g(G_2) > f_o$. The aim is to show that the A*-search never expands $G_2$ before $G$, which if it did would mean that the search terminates with a suboptimal solution.

Consider a node $n$ that is currently a node on an optimal path to $G$. As $h(n)$ is an optimistic prediction, the following must be true:

$$f_o \geq f(n)$$

Moreover, if $n$ not is expanded before $G_2$, we get

$$f(n) \geq f(G_2)$$

Combining these two gives

$$f_o \geq f(G_2)$$

But because $G_2$ is a goal state, we have $h(G_2) = 0$; hence $f(G_2) = g(G_2)$. Thus, it is proved from our assumptions that

$$f_o \geq g(G_2)$$

This contradicts the assumption that $G_2$ is suboptimal, so it must be the case that A* never selects a suboptimal goal for expansion. Hence, because the expansion of a goal node leads to the search terminates, the solution must be optimal.

As mentioned previously, the heuristic function must be optimistic, which means that the true value of the remaining path must be larger than the estimated value $h(n)$. If this was not the case, A* would not be optimal which follows from the proof.

### 2.5.3 Proof of the completeness of A*-search

Completeness of a search method, means that it always will find a solution, given that one exists, and that it terminates if no solution exists.

The fact that A*-search expands nodes in order of increasing evaluation function $f$, it must sooner or later expand the goal node. This is true under the assumption that there is a finite number of nodes (the set of states is bounded). The only two ways there could be an infinite number of nodes are

* there is a node with an infinite number of children
* there is a path with an infinite number of nodes

This means that A*-search is complete on locally finite graphs (graphs with a finite branching factor), provided there is an operator cost at least bigger than 0.

### 2.5.4 Complexity of A*-search

Although the proof is too complex for this report, it has been shown that the number of nodes expanded is exponential in the length of the solution. This is true only if the error in the heuristic function grows no faster than the logarithm of the actual path cost:

$$|h(n) - h'(n)| \leq O(\log h'(n))$$

where $h'(n)$ is the true cost of getting from node $n$ to the goal.

A good heuristic provides enormous savings of time, compared to blind search strategies. The time and space complexity for Breadth-first search is $O(b^d)$ where $b$ is the branching factor and $d$ is the depth of the shallowest solution. For Depth-first, the time complexity is $O(bm)$ and the space complexity $O(bm)$, where $m$ is the maximum depth.

As all generated nodes are kept in the memory, this is the biggest drawback of A*. Most often, A* runs out of memory before it runs out of time. However, there are modified algorithms that overcome the space problem without sacrificing optimality or completeness. These are further discussed in [Rus95].
3 The surrounding world

In order to find a mission route in a mission zone, the environment around the aircraft must be represented in some way. There are natural objects in the region, such as mountains, hills, lakes, rivers and forests that will affect any route. Man-made objects like houses or towers, and of course enemy objects (radar, anti aircraft missile sites, etc.) must also be taken into consideration.

In this chapter we will explain how the geographical map is quantified in order to search for a mission route. A homogeneous quantification is described in section 3.1, and a heterogeneous quantification in section 3.3. In section 3.2, two different ways of representing threats are described.

3.1 Quantification of the geographical map

As the aircraft can be at any geographical coordinate, up to altitudes greater than 10 000 meters, the geographical map is a continuous three dimensional set of states. Each state corresponds to a coordinate in space.
In order to search for a route in the space, the ground surface is homogeneously divided into small squares nodes which are stored into a map data base. For each node there is information about the terrain level. The terrain can be regarded as static as changes normally take a long time, so the information of the current mission zone is downloaded before the take off which saves computational power during a replanning procedure.

In figure 3.1, a region with mountains is quantified into 30x30 nodes, and the terrain height of each one is stored. In the two-dimensional discrete map, higher terrain is represented by lighter fields.

3.2 Representation of threats

A threat has some typical characteristics that constitute its representation. Besides geographical position, it has a firing range in case of anti aircraft missile site, or a cover area in case of an enemy radar. Its range is dependent on the altitude because of obstacles in its surroundings like forest, hills and buildings (see figure 3.2).

Moreover, the position data of the threat is not 100% accurate, because of uncertainties in sensor readings. A major problem is how to take the sensor noise into consideration when replanning a route. Below two ideas are described.

3.2.1 Analytic representation of threats

If the uncertainty of the sensor data is described with white gaussian noise, the threat’s position will have different errors in x and y position. The differences come from that various sensors have different accurancy in distance and angle resolution. For example, infrared sensors have good angle- but bad distance resolution, and for microwave radars the opposite is true. Data from external sensors, such as from other aircraft or ground command centrals, have their uncertainties, and all together the position of threats can be determined with quite high accurancy, but not enough to eliminate the uncertainty. For further reading about sensor management see [Fer98] and [Jen97].

By prolonging the range of the threat with three standard deviations of the noise, an elliptical area will surround the centre of the threat. This area describes in some ways the possible range of the threat, with uncertainties taken into consideration [Fab99].

In figure 3.3 the range is prolonged with $3\sigma_x$ respective $3\sigma_y$ which means that the probability of actually being too close to a threat is 99.7% inside the ellipse.

When planning a route, the pilot wants to avoid entering into a threat area. The further away from the centre, the less is the risk of the threat because of the standard deviations of position. According to that fact, a threat value is calculated under the following rules:
• The shaded part (see figure 3.3) has threat value 1
• The area described by the ellipse, gets values according to some decreasing function, ranging from 1 (the periphery of the inner circle) to 0 (the periphery of the ellipse)

The threat value corresponds to the risk in entering into a map node.

![Figure 3.4](image)

**Figure 3.4**  
Representation of threat area (A - side view, B - top view).

In figure 3.4 the graphical representation of threat area values is shown from the side and from above. One can think of the threat area as a hill - the larger the risk the higher the hill is. This “hill” is stored in the map data base, with a risk value corresponding to each map node. This can be compared to storing the height of the terrain.

### 3.2.2 HIMM representation of threats

Histogram in-motion mapping which is found in [Bro98], or simply HIMM, is a technique that upgrades the map grid (the map data base) after each sensor reading. As the map is divided into small squares, one can create a histogram that stores the number of detections of a threat in each one of these squares. The square-nodes that get the most detections, must be the most risky ones. By normalizing the histogram, the most risky nodes get the value 1, and non-risky nodes gets value 0. HIMM may be used as a complementary method to analytic representation (see previous section) or just by itself.

![Figure 3.5](image)

**Figure 3.5**  
HIMM representation of two threats.

In figure 3.5 a histogram in-motion mapping is plotted after 10, 20, 100 and 500 sensor readings. The sensor error is described by white noise which is different in x and y axis. They have a standard deviation of 2 respective 3 squares. There are two threats in the area, both with a range of 2 squares, and their position is unknown from the beginning.

After 10 sensor readings, one can discern the position of the threats, but still the distributions is quite noisy and unsmooth. It takes about a hundred iterations before a smooth and uniform histogram starts to appear, and after 500 sensor readings, the positions of the threats are easily determined to (10,10) respective (20,20). The shapes of the histograms are normal distributed, as a result of the gaussian white noise in the sensor readings.

One problem with this method, is that there is no good representation of the threats before 100-500 sensor readings, which could give inaccurate results from the mission route replanning. And as a threat tends to emit as little radiation as possible in order to not be discovered, it would take too long time to get a significant number of detections.

Another issue is how to deal with threats that for some reason disappear. With this method, it takes another 500 iterations, before a
threat is significantly removed from the map data base.

In the same way as in section 3.2.1 above, the threat position distribution functions corresponds to the risk of being in a certain node. The closer the aircraft is to the top of the “threat-hill”, the bigger is the chance of being inside of a dangerous area.

3.3 The tangent graph method

Instead of dividing the map into a homogeneous mesh (as described in section 3.1), one can search for crucial nodes in a continuous area that constitute a safe route. The tangent graph method, described in an earlier master thesis at Saab Dynamics [And94] finds crucial nodes that form a heterogeneous mesh.

In order to apply this method, the threats are analytically described by ellipses as described in section 3.2.1 above. By using the fact that between two ellipses, four mutual tangent lines can be drawn (see figure 3.6), the crucial nodes in a map can be determined.

\[
\sum_{k=2}^{2H(H+1) - 1} k
\]

In a zone with 10 threats, there will be approximately 24 000 nodes, which is not so much compared to high resolution homogenous map quantification. In the simulations of this thesis, the map is homogeneously divided into 225 000 nodes.

Figure 3.6 Two ellipses have four mutual tangent lines and five intersection points.

In an area with a number of threats, mutual tangent lines are drawn between all of the ellipses. The intersection points of all the lines are the crucial nodes (see figure 3.7).

If there are H ellipses and two start- and goal points, there are at maximum \(2H(H+1)\) mutual tangent lines, and the number of crucial nodes are

Figure 3.7 Four threats, tangent lines and intersections points.

By following the tangent lines, and only changing direction at the crucial nodes, a secure route can be found from one point to another (see figure 3.8).

Figure 3.8 An example of a route between two points.
3.3.1 Standard relations of the ellipse

In a cartesian coordinate system, an ellipse is described by
\[
\left(\frac{x-x_0}{a}\right)^2 + \left(\frac{y-y_0}{b}\right)^2 = 1
\]
where \((x_0,y_0)\) are the centre coordinates and respectively \(a\) and \(b\) are the axis length in x- and y-direction. In parametric form the same ellipse is described by
\[
x = x_0 + a\cos\theta \\
y = y_0 + b\sin\theta
\]
where \(\theta\) goes from 0 to \(2\pi\).

The slope of the tangent line of the ellipse in some point on the periphery is dependent of the angle parameter \(\theta\), and its equation is derived by differentiating \(x\) and \(y\) in the parametric equations with respect to \(\theta\).
\[
\frac{dx}{d\theta} = -a\sin\theta \\
\frac{dy}{d\theta} = b\cos\theta
\]

The straight line equation of the tangent line is
\[
y = kx + m \Rightarrow y = \frac{b\cos\theta}{-a\sin\theta}x + y_0 + \frac{b\cos\theta}{a\sin\theta}(x_0 + a\cos\theta)
\]
where \(m\) is solved from the slope \(k\) and an arbitrary point \((x_0,y_0)\) on the periphery of the ellipse. In figure 3.9 the standard relations of an ellipse is shown graphically.

![Figure 3.9 The relations of an ellipse.](image)

3.3.2 Finding the intersection points

In order to find mutual tangent lines for two ellipses, one must solve for which points \((x_{01},y_{01})\) and \((x_{02},y_{02})\) on the two ellipses the following equations are valid:
\[
k_1 = k_2 \\
m_1 = m_2
\]
That is, the tangent lines of two ellipses are mutual, when their straight line equations are equal.

Before solving the equation above, the problem is simplified by translating the plane such that one of the ellipses is transformed into the unit circle. The new coordinates are
\[
x' = \frac{x-x_{01}}{a_1} \Rightarrow x = a_1x' + x_{01} \\
y' = \frac{y-y_{01}}{b_1} \Rightarrow y = b_1y' + y_{01}
\]
\[
a' = \frac{a}{a_1} \\
b' = \frac{b}{b_1}
\]
where \(x_{01}, y_{01}, a_1\) and \(b_1\) describes the relation of the first ellipse. In figure 3.10 a transformation of two ellipses is shown.

![Figure 3.10 Transformation of ellipses.](image)
As a result of the transformation, the tangent line of the first ellipse (that is now the unit circle) can be written
\[ y_1' = \frac{\cos \theta_1}{\sin \theta_1} x_1' + \sin \theta_1 + \frac{\cos \theta_1}{\sin \theta_1} (\cos \theta_1) = \frac{\cos \theta_1}{\sin \theta_1} x_1' + 1 \]

The following equation system must be solved in order to find the mutual tangent lines
\[
\begin{align*}
    k_1 &= k_2 \\
    m_1 &= m_2
\end{align*}
\]
\[
\begin{align*}
    \frac{b_2'}{a_2} \tan \theta_1 &= \tan \theta_2 + n \pi \\
    \frac{1}{\sin \theta_1} &= y_{02'} + b_2' \sin \theta_2 + \frac{b_2' \cos \theta_2}{a_2} (x_{02'} + a_2' \cos \theta_2)
\end{align*}
\]

As there are always four mutual tangent lines for two ellipses, four pairs of angles will be the answer of the equation system. These angles describe where the mutual tangent lines strike the ellipses.

Due to the complexity of the equations, no analytic solution to the system has been found in this thesis. Computerized equation solvers like MathCad\(^1\) has been tried out with no success. With the help from a numerical method, the angles can be found though.

Once the four pairs of angles have been solved for, an inverse transformation is performed to get the tangent points in the original coordinate system. Notice that this equation needs to be iterated for each pair of angles \( \theta_1 \) and \( \theta_2 \).
\[
\begin{align*}
    x_2 &= a_1 x_2' + x_{01} = a_1 (x_{02'} + a_2' \cos \theta_2) + x_{01} \\
    y_2 &= b_1 y_2' + y_{01} = b_1 (y_{02'} + b_2' \sin \theta_2) + y_{01} \\
    x_1 &= a_1 x_1' + x_{01} = a_1 \cos \theta_1 + x_{01} \\
    y_1 &= b_1 y_1' + y_{01} = b_1 \sin \theta_1 + y_{01}
\end{align*}
\]

In figure 3.11, the four pairs or angles are plotted for the two ellipses. These points were calculated in MATLAB, with the help from an iteration procedure.

---

1. MathCad is a computer program used to analytically solve mathematical equations.
The tangent graph method

3.3.4 Advantages and disadvantages of the tangent graph method

The fact that the number of nodes decreases by using the tangent graph method has its advantages and disadvantages. A small number of nodes saves storage space in a map database, and thus searching for a route is faster.

The problem is, as the number of nodes decreases, i.e. the resolution is lower, the result of the route searching algorithm is deteriorated. As can be seen in figure 3.8, the shortest route is not found because of the zigzag route that appears. Another point is that this method is only evaluated in a 2-dimensional set of states. Problem may occur when dealing with a 3-dimensional space, as one has to deal with mutual tangent surfaces rather than mutual tangent lines.

4 Improvement of A*-search

When applying the A*-search on a homogenous map grid with square nodes, the result tends to be a curvy route with many waypoints. This requires a second treatment to achieve a proper route.

In figure 4.1 a simple route finding example is presented. The map is divided into 10x10 nodes, and from the upper left corner marked S a route is to be found to the node marked with a G. The route should be as short as possible, but the threat in the centre part of the map must be avoided.

From each node, the A*-search can move to any neighbouring node in directions north, east, south and west. The evaluation function \( g(n) + h(n) \) puts cost to distance travelled from the start node to node \( n \), plus the straight line distance from node \( n \) to the goal node. Moreover, \( g(n) \) puts an extra cost of risk to nodes that contain a threat.

As the resolution is very low, the resulting route consists of numerous turns. A verbal description of the route would be: *first go one step west, then one more step west, another step west, and one more to west. Then one step south - one step west - one step south... and so on.*
No pilot would really like to follow such a route with all its turns and small steps. Clearly there is a shorter route, simply: go 6.3 length units in direction 108 degrees, and then 6.7 length units in 153 degrees. There is the goal! (see figure 4.2).

Figure 4.2  A smarter route described by two straight lines.

The problem of a curvy route is a result of quantifying the map and the operators. An increase of the resolution may result in a better route, but slows down the algorithm time and consumes a lot of computer memory.

In this thesis two methods are presented to by-pass this problem. The first is to perform a second treatment of the route, which is called node-reduction path refinement (see section 4.1). In the example above, a reduction from 18 to 3 nodes, including the start and goal nodes, is performed. The second method, called dynamic A*-search, is an approach to use dynamic operators that works in a continuous set of states (see section 4.2).

4.1 Node-reduction path refinement

The following section builds on an idea presented in a report from NASA, in which the problem of path finding of planetary rovers are treated [Sno99].

From chapter 2 the reader is familiar with that A*-search builds up a search tree during its performance. The root of the tree is the start node, and from that point there is a path down to the node which corresponds to the target point in the map. When the search algorithm has found the goal, the solution path is stored in a dynamic list for further treatment by the node-reduction path refinement. Dynamic lists are abstract data types, and for further reading see [Ska96]. In figure 4.3 there is a schematic picture of such a list, with the coordinates of the solution path (gray squares) in figure 4.2 stored in each element.

Figure 4.3  Dynamic list of length N with coordinates of the solution path.

The node-reduction path refinement works in the following way: starting with the best option, that is a straight line between the start and goal nodes, an iterative back tracking is performed. If the straight line between node 1 and N is obstacle free, the route is done and optimal. Otherwise, one examines the straight line between node 1 and N-1, to see if that one is obstacle free. This is repeated until an obstacle free straight line is found between node 1 and node p (see figure 4.4).

Figure 4.4  Searching for straight lines that are obstacle free.
When an obstacle free straight line is found, the steps above are repeated, this time starting with a straight line between node p and N. If the goal (node N) can be reached at once, the solution is found, otherwise the backtracking procedure is repeated. In figure 4.5 the pseudo code of the node-reduction path refinement algorithm is presented.

```
q=1; //List reference = 1
p=N; //length of list of solution path

while(q != N)
{
    while(p-q != 1)
    {
        if(free sight between node(q) and node(p))
        {
            add node(q) as last element in NEW_LIST
            if(p==N) //Target node connected with line
            {
                add node(p) as last element in NEW_LIST
                return NEW_LIST
            }
            else
            {
                q=p //List reference moved forward to p
                p=N
            }
        }
        else
        {
            p=p-1
        }
    }
    q=q+1; //List reference moved forward to q+1
    p=N;
}
add node(N) as last element in NEW_LIST
return NEW_LIST
```

Figure 4.5 Pseudo code for node-reduction path refinement algorithm.

### 4.2 Dynamic A*-search

Dividing the map into small uniform squares, and applying A*-search on the mapgrid, is a very static way of searching for a route. Say, for example, that there is only one threat in the zone, and a way around it is to be found. Starting in a point far away from the threat, it is not necessary to advance in small steps. Closing up to the threat, that is being in a critical area, small steps must be taken, but when moving away from the threat larger steps can be taken again.

This pattern with different step lengths makes it necessary to leave the static operators (going from one node to a neighbour node), and let an operator step be arbitrary long, pointing in an arbitrary direction.

![Dynamic A*-search](image)

In figure 4.6 an example of dynamic operators is shown. To the left, the operators of A*-search is bounded to the square nodes, and only small steps can be taken in eight different directions. To the right, the steps have different length, depending on how close to the threat the search is. The angles are not bounded to only eight directions, but is continuous. We can say that the quantified set of states is abandoned, and a continuous set of states is used.

In section 7.3, Quantification drawbacks, it is proven that quantified operators causes routes that are not optimal. With dynamic operators, this problem is by-passed, which means that dynamic A*-search does not need a second treatment (for example with node-reduction path refinement).
4.2.1 Step length

A step can be arbitrary long, and is not quantified. Close to a threat, the steps shall be small, otherwise they shall be long. Another point is that the step should not be longer than the remaining way to the target point, otherwise there is a risk that the A*-search never finds the goal.

In figure 4.7 the basic idea of calculating the step length is shown: Being in some position in the map (represented by the small spots), the distance to the periphery of the closest threat area is the step length of the operator. If the target point is closer than any threat, then the distance to that one is the operator’s step length.

A lower limit $l_{\text{min}}$ of the step must be set, otherwise there is an infinite number of operators when approaching a threat (the step length becomes infinitesimal), and this interferes with the completeness of A*-search (see section 2.5.3).

4.2.2 Turning sector

The turning sector is dependent on the step length. A big step, permits the turning sector to be large, and for small step only small turns are permitted. This is in some way reflecting the reality: an aircraft cannot make too sharp turns. An appropriate relation between the step length and the turning sector is:

$$\theta = \frac{\theta_{\text{max}} - \theta_{\min}}{l_{\text{max}} - l_{\text{min}}} \cdot (l - l_{\text{min}}) + \theta_{\min} \quad l_{\text{min}} \leq l \leq l_{\text{max}}$$

$$\theta = \theta_{\text{max}} \quad l \geq l_{\text{max}}$$

where $\theta_{\text{max}}$ respective $\theta_{\min}$ are the turning sector limits, and $l_{\text{max}}$ is a constant that decides for which step length the maximal turn can be achieved. In figure 4.8 the graph of the function is plotted.

Saying that the turning angle is continuous in the turning sector, is not the whole truth. Even if the aircraft is permitted to point in any direction, only a few number of turns are applied as operators. A discrete number of steps in a few angles of the sector are performed in order to change state (see figure 4.9). If not a discrete number of angles was selected, the completeness criteria would be violated (see section 2.5.3).
5 Iterative improvement

Instead of finding a route with A*-search, and then reduce the number of turns (nodes), another approach is to decide the maximum number of waypoints allowed in a mission route, and then perform an optimization process in order to find a preferable route. The general idea is to start with a complete configuration of waypoints, and make modifications to improve its quality. In this chapter, a few common iterative improvement algorithms will be presented.

Let us assume that a good route does not contain more than four waypoints including the start and goal nodes, and that a polygon leg cannot be shorter than 2 length units. In figure 5.1, three different routes that follow these restrictions are plotted. Of course there is a large number of routes, and some of them are better and some of them are worse.

![Figure 5.1](image)

In order to optimize the route, it must be possible to evaluate its cost. A relevant cost would be, as the goal is to find the shortest, threat free...
way, to add the total distance and the amount of the route that cuts a threat:

\[ E_{\text{cost}} = L_{\text{route}} + \alpha L_{\text{threat}} \]

The cost of passing through a threat area, can be adjusted with the parameter \( \alpha > 0 \). In figure 5.1, the costs for the three routes are shown for \( \alpha=1 \) respective \( \alpha=10 \). The only route affected by the parameter \( \alpha \) is the one that crosses the threat area.

None of the routes in figure 5.1 are optimal, but by moving the waypoints in a smart way, the cost function will decrease and finally reach a minimal value. According to the evaluation function, a minimal value is desirable as the idea is to find a short and risk free route.

Any configuration of waypoints is called a state, and in order to change into another state, one must define an operator. Recalling from section 2.1, an operator is an operation that changes a state into another state. One possible operator could be moving one of the internal waypoints between the start and goal nodes, one step north, east, south or west.

### 5.1 Hill-climbing search algorithm

Hill climbing is an iterative improvement algorithm, described in [Rus95]. Starting with a configuration of waypoints (the initial state), an iterative procedure changes the state in each step, in such a way that the cost function of the route decreases. In figure 5.2 the pseudo code of the Hill-climbing algorithm is presented.

#### Pseudo code for Hill-climbing algorithm

**Input data:** Initial state (configuration of waypoints)
**Output data:** Optimal state (configuration of waypoints that gives the optimal route)
**Variables:** CURRENT_STATE, NEXT_STATE, CHEAPEST_STATE

\[
\text{CURRENT}_{-}\text{STATE} = \text{initial state} \\
\text{LOOP} \\
\quad \text{for all operators} \\
\quad \quad \text{NEXT}_{-}\text{STATE}_{i} = \text{operator}_{i}(\text{CURRENT}_{-}\text{STATE}) \\
\quad \quad \text{CHEAPEST}_{-}\text{STATE}_{-}\text{COST} \leftarrow \text{min}(\text{cost}(\text{NEXT}_{-}\text{STATE}_{i})) \\
\quad \quad \text{if } (\text{CHEAPEST}_{-}\text{STATE}_{-}\text{COST} < \text{cost}(\text{CURRENT}_{-}\text{STATE})) \\
\quad \quad \quad \text{CURRENT}_{-}\text{STATE} = \text{NEXT}_{-}\text{STATE}_{i}, \text{min} \\
\quad \quad \text{else} \\
\quad \quad \quad \text{return CURRENT}_{-}\text{STATE} \\
\text{end LOOP}
\]

*Figure 5.2 Pseudo code for Hill-climbing search*

### 5.2 Simulated annealing

For a route with four waypoints, including the start and goal node, there are eight different operators, as there are two internal waypoints that each one can be moved in four directions. All of these operators are applied, and the one that decreases the cost function most is chosen. In that way the algorithm changes the waypoints into a better configuration in each step, aiming for the global minima of the cost function.

One problem with this strategy, is that the algorithm may get stuck in a local minima, and return a suboptimal route. One way to get around this, is to choose only one operator by random, and apply that one if it decreases the cost of the route. With such a strategy, the algorithm does not move towards the optimal value directly, and may in that way avoid ending up in local minimas.

Another way to find the global minimum, is to let the initial state change from time to time. **Random restart hill-climbing** does just this: it conducts a series of Hill-climbing searches from randomly generated initial states, running each until it halts or makes no discernible process. It saves the best result found so far from any of the searches.

Still the problem with local minimas remains, and there are other methods that use less strict requirements of always changing into a better state. One of them is Simulated annealing.

**Simulated annealing**

Instead of starting again from the beginning when stuck in a local minima (as Random restart hill-climbing), the search can move to a worse state to escape the local minima. This is the idea of Simulated annealing, which is described in [Rus95]. The loop is quite similar to Hill-climbing search, but instead of picking the best operator, it picks a random one.

If the operator decreases the cost function, that is a better set of waypoints is found, it immediately chooses this state. Otherwise, if the cost function increases with \( \Delta E \), it chooses this worse state with a probability less than 1. The probability is inversely proportional to \( \Delta E \), and also dependent on a parameter \( T \). \( T \) in its turn, is mapped by some decreasing function, dependent on the number of cycles of the algorithm loop. When \( T \) approaches zero, the probability of choosing worse states decreases, and finally Simulated annealing almost behave like Hill-climbing search. However, in the beginning of the process, it chooses worse steps more often, and therefore tends not to get stuck in local minimas. In figure 5.3 the pseudo code of Simulated annealing is
Simulated annealing

Input data: Initial state
Output data: Optimal state
Variables: \( T \) - decreasing variable
\( CURRENT\_STATE, NEXT\_STATE \)

for (\( t = 1 \) to INFINITY)
{
    \( T = \text{sche}u\text{d}e\{t\} \)
    if (\( T == 0 \))
        return \( CURRENT\_STATE \)
    \( NEXT\_STATE = \text{o}\text{p}\text{era}t\text{or}_{\text{random}} \{CURRENT\_STATE\} \)
    \( \Delta E = \text{cost} \{CURRENT\_STATE\} - \text{cost} \{NEXT\_STATE\} \)
    if (\( \Delta E > 0 \))
        \( CURRENT\_STATE = NEXT\_STATE \)
    else
        \( CURRENT\_STATE = NEXT\_STATE \) with probability \( e^{\Delta E/T} \)
}

Figure 5.3 Pseudo code for Simulated annealing.

Simulated annealing copies the way that nature itself reaches global minima: When a liquid is gradually cooling until it freezes, its total energy of the atoms in the material \( E \) approaches the minimum level as the temperature \( T \) decreases. During the cooling process, random thermal noise may increase the total energy temporarily, which corresponds to the deteriorating step in the Simulated annealing algorithm.

One can prove, that if the temperature \( T \) is lowered slowly enough, the material will reach the lowest energy \( E \). That is, if the schedule procedure in the algorithm above lowers \( T \) slowly enough, an optimal configuration of waypoints will be achieved.

6 Mission route planning

In the simulations of this thesis, the following procedures are used to solve the mission route planning problem:

- Homogeneous quantification of the map (section 3.1),
- Analytic representation of threats (section 3.2.1),
- A*-search (section 2.5) and
- Node-reduction path refinement (section 4.1).

A homogeneous mapgrid is a straightforward quantification and easy to deal with for different search methods. The analytic representation of threats is faster than HIMM-representation, and easy to work with in the simulations. A*-search and node-reduction path refinement are the main algorithms of the simulations, but also iterative improvement algorithms, Greedy search and dynamic A*-search (without node reduction) are treated and evaluated.

The four steps above are sequentially executed, and in figure 6.1 a blackbox model of the entire replanning system is shown.

In this chapter the algorithms are refined and modified, in order to apply them to mission route planning problems for military aircraft. An interview with people that have relevant experience in the area, gives valuable information of how to adjust the algorithms in the right way.
6.1 Interview

To be able to refine the system, an interview with Robert Johnsson and Bengt Larsson at Saab Aerospace in Linköping was done. Robert Johnsson has been working at FMV Prov with areas concerning test flights of military aircraft. At Saab he is working with HMI (Human Machine Interface). Bengt Larsson has been working as a pilot for 35 years, flying JA-35 Draken and JA-37 Viggen. At Saab he is working with Tactical operational analysis.

The following are the questions that were asked, and a summary of the answers during the interview:

1a. How important are predefined waypoints? Does the pilot wish to keep them, or could they be placed arbitrary in a replanning situation?

Predefined waypoints are important, as it makes it easier for the pilot to memorize them. If there is some waypoint that the pilot wants to keep, he should be able to point it out and override the replanning system.

1b. Is it important to place the new waypoints at landmarks that are easy to recognize?

It is not crucial to place them at landmarks, most important is a safe route. Moreover, if the replanning system has located other landmarks than the pilot knows about, they would be useless for memorisation.

2. In which way, and how much do the following factors affect when planning a mission route: threats, terrain, turns at waypoints, number of waypoints, fuel, weather situation, time restriction, altitude and bearing at goal point.

The threat situation is the most important factor. The mission route is planned around this, and the other factors are more like second conditions.

High terrain forces the pilot to increase altitude, which is not desirable. On the other hand, terrain may be a way to avoid a threat or hide from it.

Turns can also be used to avoid a threat, but cannot be sharper than the MLL (Maximum Load Level) allows. The MLL is an upper limit of the forces that are permitted to affect the aircraft in a turn, and it is dependent on the load weight. At an escape situation, the load of the aircraft is most probably released, so sharper turns can be performed. However, turns are normally not that important.

The number of waypoints is not so important, because the route is described by the polygon legs. For a large number of waypoints, the polygon leg is described by a curve rather than by a straight line. This is no problem, as the aircraft can follow a route by coupled steering. There are cases though, where waypoints are necessary. For example, at target points, checkpoints and points where some change of the aircraft is performed, waypoints are used.

Fuel affects the route only if it runs out. That means, that any route is good concerning fuel, if there is enough fuel left at the goal point to fulfil the mission and get back to a base. If, in a replanning situation, no route exists that keeps the fuel margins inside of the safety limits, the mission should immediately be aborted.

Weather situation should not be concerned during a replanning. It is assumed that the pilot will be able to handle bad weather situations. Maybe a minimum altitude can be added as a restriction, if there is much varying terrain and bad sight.

Time restrictions are not important as far as the pilot keeps inside of the time window of the mission. If, for example, the new route would be longer than the preplanned route, the pilot will have to increase speed. If it is impossible to arrive at the goal point at the right time, the mission shall be aborted.

Altitude is important, as it is harder to discover an aircraft at low altitude. Terrain serves as a protection from radar and missile launchers, so it is desirable to keep low. If the pilot is forced to increase the altitude, the risk of the mission increases.

Finally, the bearing at the target point affects the replanning, but is not that important. In case of unpredictable situations, the bearing can be redefined at a later moment.

3. How short can the polygon legs be?

It does not matter how short the legs are, as many short straight lines describe a curve and this is acceptable. The only importance is that the turn of the curve is not sharper than the MLL.
Interview

4. Does the pilot ever fly inside of a risk area (according to threats), or does he rather abort the mission than be exposed to a risk?

It depends on the risk threshold, which is dependent on the importance of the mission. A very important mission, means a high threshold and therefore the pilot could choose to enter into a risky area.

5. Under which circumstances is a mission aborted?

A mission is aborted when the risk threshold is exceeded, there is not enough fuel or the pilot exceeds the time requirements.

6. In a replanning situation, does the pilot keep far away from the threats, or does he sneak around them?

If it is possible, the pilot adds a safety margin in order to stay away from risky areas. Though sometimes, when the threats are close together, the pilot may have to sneak around the threat periphery.

7. In which way should the system present and execute the rerouting?

When the predefined route exceeds the risk threshold, the system should automatically perform a replanning. The pilot will be notified that there is a new and safer route, and the proposal shall be presented graphically on the displays. The pilot will then choose to execute the new route, or edit some of its waypoints. If the edited route is approved according to the threshold, the pilot can choose to execute it, search for alternative routes or just stick to the predefined one. The most important thing is, that the pilot has a dialogue with the system in order to find a satisfactory route. The replanning system is never allowed to execute a new route by itself, without the approval of the pilot (see [Sch98]).

Refinement of algorithms

6.2 Refinement of algorithms

From the interview the following refinements are made to the algorithms and representations described in the sections 2.5, 3.2.1 and 4.1.

6.2.1 Evaluation function of A*-search

From the interview we make the conclusion that the risk which is connected to a route, is a good way to evaluate the cost. As mentioned in questions 4 and 5, the risk is all the time compared to a risk threshold which is not to be exceeded. Therefore a natural choice of cost function is the risk of going from one state to another.

Passing through a threat area, which is described as a risk hill (see section 3.2.1), will add a risk value to the evaluation function. Also if the pilot needs to increase altitude, because of terrain or other factors, risk will be added to the evaluation function. Finally, as the pilot wants to take the fastest route possible, the distance travelled also contributes with a cost. This cost can also be considered as a risk cost, as staying in the air above a battle field corresponds to a risk. The cost function used in the simulations of this thesis is:

$$g(n) = D(n) + \alpha H(n) + \beta R(n)$$

where $D$ is the distance travelled, $H$ is the height of terrain and $R$ is the threat value. $\alpha$ and $\beta$ are constants which put different weights to $H$ and $R$ depending on their importance.

The heuristic function $h(n)$ is simply the straight line distance from node $n$ to the goal node. That means that if there are no threats or high terrain in the area, the straight line route will be the best. That is a natural choice for an aircraft. As a summary, the evaluation function used in this thesis is:

$$f(n) = g(n) + h(n) = D(n) + \alpha H(n) + \beta R(n) + SLD(n, n_g)$$

where $SLD(n, n_g)$ is the straight line distance between node $n$ and the goal node. $\alpha$ and $\beta$ are experimentally calibrated to get satisfactory results.

6.2.2 Safety margin

In question 6 in section 6.1, it is mentioned that an extra safety margin around threats is added when planning a route. In order to take this demand into consideration, the major and the minor axis of the threat ellipse are extended by the size of the safety margin.

The elliptic threat area around a threat at position $(x_0, y_0)$ is risk
Refinement of algorithms

weighted due to a decreasing function (see section 3.2.1 and section 6.2.3). The ellipse is hereby described as:

\[
\frac{(x-x_0)^2}{r+3\sigma_x+e} + \frac{(y-y_0)^2}{r+3\sigma_y+e} = 1
\]

In figure 6.2, an extra margin \(e = 4\) kilometres is added to the threat area (compare with figure 3.3), and this is the value used in the simulations.

6.2.3 Decreasing risk function of threat area

As mentioned in section 3.2.1 the area outside the threats range, but inside of the threat ellipse shall be weighted with a decreasing function dependent on the distance from the threat’s position. Below is the decreasing function, that is used in the simulations of this thesis.

\[
1 - \frac{\left(\frac{x-x_0}{R_x^2} + \frac{y-y_0}{R_y^2}\right)}{1 - \left(\frac{r}{\max(R_x, R_y)}\right)^2} \quad \forall \quad r \leq x \leq r+3\sigma_x+e \\
\quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad r \leq y \leq r+3\sigma_y+e
\]

where \(R_x = r+3\sigma_x+e+m\) and \(R_y = r+3\sigma_y+e+m\). In figure 6.3 the risk hill is plotted for a threat. The additional value \(m\) is set to a value larger then 0, and is used to lower the number of waypoints. This is further described in section 6.3.1 below.

Additional algorithms

6.3 Additional algorithms

From question number 2 in section 6.1 it is clear that the number of waypoints, fuel consumption and a predefined approach bearing may affect the choice of a route. These factors are added as secondary algorithms, in order to evaluate their influence on route planning.

6.3.1 Lowering the number of waypoints

Sometimes the pilot may wish to have a small number of waypoints, with connecting straight lines, plotted in the map display. A*-search does not take this requirement into consideration, but with node reduction refinement “unnecessary” nodes are cancelled. Still the number of waypoints are very large, because of circular segments around the threat areas.

By adding an extra margin to the threat range and terrain edges, and allowing the node reduction refinement to cross a threat area with a low risk value, the number of waypoints are significantly lowered as described below. The new threat range will be:

\[
R_x = r+3\sigma_x+e+m \quad \text{and} \quad R_y = r+3\sigma_y+e+m
\]

where \(m\) is the extra margin. This range is used to value the risk inside of a threat area (see section 6.2.3).

In figure 6.4 a simplified picture of the benefit of extra margin is shown. To the left, no extra margin is added, and the node reduction refinement procedure manage to reduce all nodes on a straight line from the start/goal node to the threat periphery. But on the circular segment of
the path, there are no straight lines, and no node reduction is possible.

To the right, the extra margin around the threat forces the A*-search to keep a distance to the periphery. It has the same shape as in the left picture though. The trick is to let the node-reduction path refinement pass through a threat area with risk values lower than some threshold value $T_r$. One can say that it passes over a fictitious threat area, that is actually not dangerous. The result is a smaller number of waypoints, as can be seen in the right picture in figure 6.4.

### 6.3.2 Fuel consumption

In each step of the A*-search, the fuel decreases depending on the length of the step and the attitude of the aircraft. A simple model of the fuel consumption is added, in order to examine how suboptimal routes are chosen in case there is not so much fuel left. The model of the fuel consumption is (where $k_1$ and $k_2$ are adjustable parameters):

$$
Consumption_{fuel} = k_1 \cdot \text{LENGTH} \cdot e^{k_2 \cdot \text{ATTITUDE}}
$$

If the fuel consumption is not calculated in each step, A*-search would never find a route in case of low amount of fuel. The reason of that is that A*-search is deterministic, and always chooses the same route in a specific scenario. Therefore it is not possible to first find a route, and then see if the fuel runs out or not.

By calculating the remaining fuel in each step of the A*-search, paths in the search-tree that are impossible will be pruned, and other paths that are more fuel saving will be found. The idea of pruning the A*-search is found in [Rus95].

A graphical representation of the fuel consumption is shown in figure 6.5. The values of $k_1$ and $k_2$ are used in the simulations, and are empiric.

### 6.3.3 Approach bearing

If the mission requires that the aircraft is to approach the target point with a certain bearing, this must be taken into consideration by the replanning system. By adding a fictitious terrain with infinite height to the map data base, and giving it a shape of a circle with an opening in the right direction, the A*-search is forced to find a way through this opening (see figure 6.6).
6.4 Finalizing the new route

There are two ways of finalizing the new route: Either with a low
amount of waypoints connected with straight lines, or a mission route
path described by curve segments.

Modern aircraft has coupled steering, which means that the pilot can
set automatic flight mode and let the aircraft system itself follow the
route. Then it is not crucial to have as few waypoints as possible, or to try
to set them at logical points such as at lakes, buildings etc.

Old aircraft, that do not have coupled steering, are always manoeuvred
by the pilot. In this case, a low number of waypoints is essential since it
simplifies it for the pilot.

However, from the interview it is clear that the number of waypoints
is of no importance, but one can rather see the mission route as connected
curves. Moreover, any modern aircraft has the ability of coupled steering,
why nowadays trying to get as low number of waypoints as possible, is
rather a bad restriction than an improvement of the system.

In the simulations of this thesis, the route will be finalized and
presented as one curve, from the start to the target point (see figure 6.7).

Figure 6.7 Presentation of the mission route.

Threats are described by ellipses, with the range within the solid circle
and the extended threat range by the dashed ellipses. Terrain is described
by level curves, and can be seen as hills or mountains that rises extremely
much compared to the other terrain.

6.5 Algorithm description

The following section gives the pseudo code for the subsystems in the
replanning system in figure 6.1.

6.5.1 Quantification of the map

A map data base is constructed in order to store information about each
node. For a region divided into HxW nodes, a static array of H*W cells
is constructed. Each cell can store terrain height and risk value, and has
a visit flag which is set true if the A*-search has visited it. Each node has
a coordinate, and in order to extract information from node (i,j), one finds
the corresponding array cell [i-1 + H*(i-1)]. I.e. instead of working with a
data matrix, a one-dimensional vector is addressed when saving or
loading information.

6.5.2 Putting threats to map data base

In order to take threats into consideration, each node in the quantified
map (see section 6.5.1) is weighted according to its threat (or risk) level.
In figure 6.8 the pseudo code of the threat weighting algorithm is shown.

| Input data: | x,y - threat position | range - threat range |
| Variables: | xD,yD - position standard deviations | CORRIDOR - Extra margin around threat |
| riskValue - The riskvalue of a map node |

\[
\text{boxX} = \text{range} + 3 \times \text{xD} + \text{CORRIDOR} + m \\
\text{boxY} = \text{range} + 3 \times \text{yD} + \text{CORRIDOR} + m \\
\text{for} \ i = x-\text{boxX} \ \text{to} \ x+\text{boxX} \\
\quad \text{for} \ j = y-\text{boxY} \ \text{to} \ y+\text{boxY} \\
\quad \text{ellipse} = ((i-x)/\text{boxX})^2 + ((j-y)/\text{boxY})^2; \\
\quad \text{if} \ (\text{distance}(x,y,i,j) \leq \text{range}) \\
\quad \quad \text{riskValue} = 1; \\
\quad \quad \text{else if} \ (\text{distance}(x,y,i,j) > \text{range} \ & \text{ellipse} \leq 1) \\
\quad \quad \quad \text{riskValue} = \text{decFunc}(x,y,i,j,\text{boxX},\text{boxY},\text{range}); \\
\quad \quad \quad \text{put riskValue to mapNode}(i,j) \\
\]
according to the rules described in section 3.2.1. The decreasing function decFunc is described in section 6.2.3.

**6.5.3 A*-search algorithm**

The pseudo code of the A*-search algorithm used in the simulations of this thesis is found in figure 6.9. There are two checkpoint procedures, the first controlling if the heap queue is empty. That means that no more nodes have been expanded, or in other words that no solution of the route finding problem exists. The other checkpoint is if the goal is reached. If so, the solution path is returned as a dynamic list (see figure 4.2) for further treatment by node-reduction path refinement.

If the heap queue is not empty, nor the goal reached, the current node is expanded. Only children nodes that are not visited before are added to the heap queue. Their evaluation values (see section 6.2.1) and fuel consumption (section 6.3.2) are calculated. The ones with enough fuel are added to the heap queue, and the loop starts over again checking the next node in the heap queue.

![Figure 6.9 Pseudo code for A*-search algorithm](image)

---

### 7 Simulations and results

In this chapter the results of the simulations of this thesis are presented. The reader will find graphically results in section 7.1 as well as numerical results in section 7.2. In section 7.3 and 7.4, the advantages of dynamic A*-search are shown.

In the simulations of this thesis, the geographical zone is divided into 150x150 square nodes (see section 3.1). Each node has a resolution of 500 meters, which means that a territory of 75x75 km is treated. The territory has random generated terrain, described by two-dimensional cosine functions. Threats with random characteristics are placed in the territory, and the aim is to find a route between a start point at the left side of the territory, to a goal point at the right side (see figure 7.1).

![Figure 7.1 Random scenario with threats, terrain and start- and goal points.](image)

The purpose of the simulations is to apply some of the algorithms described in this thesis (see chapter 6, Mission route planning), and examine which routes are chosen by the replanning system. Random
scenarios reflect the reality, as nobody can predict a future situation. In total, 13 random scenarios will be treated, and each route is numerically evaluated.

In order to evaluate the routes that are found by A*-search, the alternative methods described in chapter 5 (Iterative improvement) and Greedy search (see section 2.4.2) are applied to the same scenarios as a comparison measure.

Dynamic A*-search is implemented and compared to A*-search with static operators. In appendix A.2 all scenarios are graphically presented, and the routes of A*-search with static operators and dynamic A*-search are plotted.

If no other is mentioned, the numerical values used in the simulations are shown in table 7.1. All values are empiric, and have been calibrated during previous test simulations.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terrain cost ($\alpha$)</td>
<td>40</td>
<td>6.2.1</td>
</tr>
<tr>
<td>Threat cost ($\beta$)</td>
<td>100</td>
<td>6.2.1</td>
</tr>
<tr>
<td>Initial amount of fuel</td>
<td>700 units</td>
<td>6.3.2</td>
</tr>
<tr>
<td>$k_1$</td>
<td>2</td>
<td>6.3.2</td>
</tr>
<tr>
<td>$k_2$</td>
<td>5</td>
<td>6.3.2</td>
</tr>
<tr>
<td>Safety margin</td>
<td>4 km</td>
<td>6.2.2</td>
</tr>
<tr>
<td>Trig value ($T_r$)</td>
<td>0</td>
<td>6.3.2</td>
</tr>
<tr>
<td>Extra margin (m)</td>
<td>0 km</td>
<td>6.3.2</td>
</tr>
<tr>
<td>Approach bearing</td>
<td>not defined</td>
<td>6.3.3</td>
</tr>
<tr>
<td>$l_{min}$</td>
<td>2 length units</td>
<td>4.2.1</td>
</tr>
<tr>
<td>$\theta_{max}$</td>
<td>90 degrees</td>
<td>4.2.2</td>
</tr>
<tr>
<td>$\theta_{min}$</td>
<td>10 degrees</td>
<td>4.2.2</td>
</tr>
<tr>
<td>Schedule function of SA</td>
<td>1-0.01*loops</td>
<td>5.2</td>
</tr>
<tr>
<td>Max internal waypoints</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 7.1 Constants used in the simulations.

In section 7.1 below, a representative selection of scenarios from appendix A.2 is shown. The aim is to point out typical characteristics of different search methods and the additional algorithms described in chapter 5 and 6.

### 7.1 Scenarios

#### 7.1.1 A*-search always avoids threats and terrain

The route avoids threats and terrain as long as there is fuel left, and thus it goes around the cluster of objects (see figure 7.2). This is a result of the evaluation function, in which terrain and threats get a high risk cost compared to distance.

#### 7.1.2 How initial amount of fuel affects the route

Starting with 700 units of fuel, the best route is going around the terrain and threats (see figure 7.3).

Lowering the initial amount of fuel, say to 600 units, the route in figure 7.3 would put the aircraft in a situation where the aircraft is out of fuel. In this case another route is presented, which is optimal for this initial amount of fuel (see figure 7.4 and section 6.3.2).
7.1.3 Random restart hill-climbing versus A*-search

In figure 7.5 and figure 7.6 the A*-search is compared to Random restart hill-climb search. In figure 7.5 Random restart hill-climb search finds a route that avoids all threats, but it is very long and far from the best. As the internal waypoints are only three, Random restart hill-climb has no chance to improve the route from this configuration.

In figure 7.6 Random restart hill-climb presents a rather good route, that more or less is the straight line route. A*-search finds the optimal route though (according to the evaluation function). In row five in the table in appendix A.1.1, we see that A*-search has evaluation function value 109 and Random restart hill-climb has a value that in average is larger.

It is also obvious, that A*-search is deterministic and Random restart hill-climb is random. A deterministic search is desirable, as it rises the reliability of the system.
7.1.4 Greedy search versus A*-search

Greedy search always tries to minimize the length of the remaining path. Therefore, it discovers too late if a route is bad. When closing to a threat or terrain, it is forced to either turn and take a longer route than necessary (see figure 7.7) or pass through mountains or threat areas (see figure 7.8).

Greedy search normally has a shorter algorithm time than A*-search (see appendix A.1.3), and sometimes it finds the same route as A*-search. However, in average Greedy search has a lack of the intelligence that A*-search shows when finding routes.
Scenarios

7.1.5 Simulated annealing versus A*-search

Simulated annealing uses random operators, and can present a good result as well as a disastrous one. In figure 7.9 a fairly good route is presented by Simulated annealing. It is going through terrain, but avoids all threats.

In figure 7.10 a too risky route is presented by the Simulated annealing, which is going right through a threat area. In this case it gets stuck between two threats, and the only way to get away from that point is to change into worse states. This is a deep local minima of the evaluation function, and even though Simulated annealing may change into worse states according to the probability, it does not in this case.

By increasing the number of loops in Simulated annealing, or in other words lower the temperature slower, the chance of exiting a bad state like the one in figure 7.10 rises. The problem is that the algorithm time (see appendix A.1.3) would be too long if T was lowered too slowly.

Finally, the random behaviour of Simulated annealing is a disadvantage, as it lowers the reliability of the replanning system.

7.1.6 Hill-climbing versus A*-search

Hill-climb search sometimes presents good routes, and sometimes bad, and it all depends on the composition of the scenario. The explanation of this is that in these simulations, the initial configuration is always the straight line between the start and the goal points. This line is divided by the internal waypoints (three in this case), which will be moved in such way that the evaluation function decreases. In other words, if the threats are placed in such way that it benefits a nearly straight line route, Hill-climb finds a good route.

The Random restart hill-climb avoids this kind of predefined initial states in order to get around problematic scenarios. Hill-climb is on the other hand deterministic.

In figure 7.11 a Hill-climb route is presented that does avoid threats but is very zig-zag compared to the route from the A*-search. In figure 7.12 a good route presented by Hill-climbing, but according to the evaluation function the best route is the one taken by A*-search, around the threat. The reason of this is that the Hill-climbing route is crossing the foot of a mountain, which is better to avoid as long as there is enough fuel.
7.1.7 Predefined approach bearing

If there is a predefined approach bearing (see section 6.3.3), the A*-search will take this into consideration. In figure 7.13, the approach bearing is set to direction north, and in figure 7.14 it is set to direction east. The best route is different in these cases.

Figure 7.11 A zig-zag route presented by Hill-climb search.

Figure 7.12 Hill-climb finds a short route which is passing over a foot of a mountain.

Figure 7.13 The predefined approach bearing is set to 0 degrees.

Figure 7.14 The predefined approach bearing is set to 90 degrees.
7.1.8 Decreasing the number of waypoints

In figure 7.15 the A*-search presents the best route, and it touches upon the periphery of the threat ellipses. The number of waypoints in this case is very large because of the smooth curves. If a small number of waypoints by some reason is wished for, the route could be presented as in figure 7.16 (see section 6.3.1). In this case \( T_r = 0.25 \) and \( m = 1 \) km.

7.2 Numerical results from simulations

Six factors have been used in order to evaluate the search methods:

- total cost of evaluation function,
- time consumed by the data algorithms,
- amount of fuel left,
- relation of the route length and the straight line distance,
- how much of the route that crosses a threat area, and
- highest terrain peak that the route passes.

In table 7.2, the average values for each search method has been calculated. All the data from the 13 scenarios is found in appendix A.1. A*-search, Greedy search and Hill-climb search are deterministic and never change their result for respective scenario. On the other hand, Simulated annealing and Random restart Hill-climb give different results as they deal with random numbers. Therefore, in their cases, each scenario is run two times in order to get more significant values.

<table>
<thead>
<tr>
<th>Evaluation function</th>
<th>A*-search</th>
<th>Greedy search</th>
<th>Hill-Climb search</th>
<th>Simulated annealing</th>
<th>Random restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm time</td>
<td>0.85 s</td>
<td>0.17 s</td>
<td>1.36 s</td>
<td>5.68 s</td>
<td>8.34 s</td>
</tr>
<tr>
<td>Fuel left</td>
<td>22%</td>
<td>23%</td>
<td>32%</td>
<td>28%</td>
<td>18%</td>
</tr>
<tr>
<td>Route length</td>
<td>34%</td>
<td>23%</td>
<td>15%</td>
<td>22%</td>
<td>41%</td>
</tr>
<tr>
<td>Threat cut</td>
<td>0%</td>
<td>4%</td>
<td>4%</td>
<td>2%</td>
<td>1%</td>
</tr>
<tr>
<td>Highest terrain peak</td>
<td>12 units</td>
<td>24 units</td>
<td>42 units</td>
<td>40 units</td>
<td>33 units</td>
</tr>
</tbody>
</table>

From table 7.2 we make the conclusions that Simulated annealing and Random restart Hill-climbing takes too much time. The algorithm time must not be longer than one or two seconds, as the system is to work in a real time environment. Moreover, four times Random restart presented routes which put the aircraft in a situation without any fuel left, which is
Quantification drawbacks

It is important to avoid threat areas, and A*-search is solely the best in this matter; it never enters a threat area! It also avoids terrain more than the other search strategies.

The route length of A*-search is second worse, and that is the prize to pay for always avoiding threats and terrain. However, as long as there is enough fuel at the goal node and the pilot is on time, it does not matter how long the route is.

Finally, the evaluation function value, shows that A*-search really is the best search method of the ones evaluated in this thesis.

7.3 Quantification drawbacks

In figure 7.17 the route presented by the A*-search (before node-reduction path refinement) is plotted as a dashed line. As can be seen, it is not optimal, which conflicts with the proof that A* is optimal.

![Figure 7.17](image_url)

*Figure 7.17* The dashed route is presented by A*-search. It has a few dips which is a result of the quantification.

The dips that occur in the route, is a result of the quantification of the map and the operators.

7.3.1 Proof of quantification drawbacks

In figure 7.18, a simple case of the quantification drawbacks is shown. The map is homogeneously divided into discrete nodes, and a few of them are plotted in the figure. If there was no quantification of the map and the operators, the best step would be to go immediately from the current node to the goal node.

However, as there are only 8 different operators (north, north east, east and so on), and a step is only to a neighbour node, problems occur. From the case in figure 7.18, we see that according to the evaluation function \( f(n) = g(n) + h(n) = \text{step length} + \text{SLD(goal node)} \), one of the two operators must give the lowest cost, but it is difficult to say which one. The aim of the proof, is to show that depending on the relation between \( a \) and \( b \) in figure 7.18, the best operator switches.

![Figure 7.18](image_url)

*Figure 7.18* Two operators that are almost equally good.

**PROBLEM:** Solve for which \( a \), the cost of going to node 1 is lower than going to node 2.

**PROOF:** Say that the evaluation function is defined as \( f(n) = g(n) + h(n) = \text{step length} + \text{SLD(goal node)} \), then the cost functions of the two operators are:

\[
\begin{align*}
    f_1 &= 1 + \sqrt{a^2 + (b + 1)^2} \\
    f_2 &= \sqrt{2} + \sqrt{a^2 + b^2}
\end{align*}
\]
The following equation is to be solved with respect to \( a \):

\[
\frac{1 + \sqrt{a^2 + (b + 1)^2}}{\sqrt{2} + \sqrt{a^2 + b^2}} \leq \frac{2b^2 + 2\sqrt{2} - 1}{\sqrt{a^2 + b^2}}
\]

which gives:

\[
a \geq \frac{2b^2 + 2\sqrt{2} - 1}{\sqrt{a^2 + b^2}}
\]

When \( a \) is equal to the right expression, both operators are equally good. If \( a \) is bigger, it is better to go straight ahead, and if it is smaller than the right expression it is better to turn down towards the goal node.

This explains the dips in the route presented in figure 7.17 by A*-search. The node-reduction path refinement cancels these dips, and improves the route. In figure 7.17 the resulting route (full line) is plotted, and the dips are avoided.

### 7.4 Dynamic A*-search

The route found by dynamic A*-search is not treated with node-reduction path refinement, but finds proper routes anyway. Node reduction path refinement is unnecessary in this case, as the steps from the beginning are as long as possible.

In figure 7.19, two routes are plotted in a scenario with six threats. The dashed route is the one presented by dynamic A*-search. Notice that the turns are smooth, in opposite to the static A*-search route (full line). The latter route also suffer the consequences of the map quantification (see previous section), and runs into a threat before passing around.

On the other hand, sometimes A*-search static finds shorter routes, as in figure 7.20. The question is if the turn is not too sharp between the two cluster of threats (marked with an arrow). If so, the dynamic A*-search finds a route that is optimal and has smooth turns.

In figure 7.20, two routes are plotted in a scenario with six threats. The dashed route is the one presented by dynamic A*-search. Notice that the turns are smooth, in opposite to the static A*-search route (full line). The latter route also suffer the consequences of the map quantification (see previous section), and runs into a threat before passing around.

On the other hand, sometimes A*-search static finds shorter routes, as in figure 7.20. The question is if the turn is not too sharp between the two cluster of threats (marked with an arrow). If so, the dynamic A*-search finds a route that is optimal and has smooth turns.

#### 7.4.1 Visited nodes

One benefit with dynamic operators, is that it saves computer time and memory. A*-search with static operators needs to explore at least every node on a nearly straight line between the start and the target point. Using dynamic operators, makes it possible to take large steps and avoid to explore “unnecessary” nodes.

In figure 7.21, all visited nodes are plotted for respective A*-search static and A*-search dynamic. In the upper figure, all nodes on the edge of the lump of visited nodes, has the same evaluation function value. All nodes inside, must have a lower value, according to the proof in section 2.5.2 Proof of the optimality of A*-search.
In the figure at the bottom, the visited nodes of A*-search dynamic is plotted. The typical fan-shaped pattern of operators (visited nodes) is evident, and gives a plain idea of how dynamic operators work.

7.4.2 Time and memory complexity

Dynamic operators work well when there is a low number of threats in the zone. As can be seen in figure 7.22, the time and memory complexity of dynamic A*-search grows exponentially with the number of threats, while static A*-search is more or less static.

In part C in figure 7.22, the relation between the number of nodes visited and the algorithm time is plotted, and it is nearly linear. In order to save time, effort should be put in decreasing the number of visited nodes with the help of smarter operators. Dynamic operators is one approach, and an improvement compared to static operators, but is too slow if the number of threats increases.

Figure 7.22  A - Memory complexity as a function of number of threats.
B - Time complexity as a function of number of threats.
C - Relation between visited nodes and algorithm time.
Simulations of mission route planning scenarios, show that A*-search followed by node-reduction path refinement find suitable mission routes. This statement is based on 13 random scenarios on which A*-search and other alternative algorithms have been applied.

According to table 7.2 and the simulations, A*-search always finds a route within the fuel limits, it never cuts any threat areas and it is fast enough to be implemented in a realtime system. Moreover, A*-search is deterministic and easily adjusted in order to consider dynamic threats and other dynamic objects in the mission zone.

Figure 8.1 New mission route presented.

In table 1.1 (section 1.2), there are a various range of requirements and factors that affect the route planning. Taking a look at figure 8.1, and checking the factors in table 1.1, the following has been taken into consideration in this thesis:
• Enemy objects  The replanning system gives high priority to hostile objects when searching for a new route. Uncertainty in sensor data is taken into consideration, in such a way that threat areas are described by ellipses representing depth- and side standard deviations of sensor signals.

• Terrain  In the simulations, random terrain is used to represent the world around. The replanning system avoids terrain as far as possible, as increasing altitude increases the risk of the mission.

• Fuel  The system calculates the fuel consumption of a route, and when there is too little fuel left for any route, the mission is aborted.

• Approach bearing  If there is a predefined approach bearing, the replanning system takes this into consideration.

• Waypoints  A low number of waypoints is optional, but according to the interview it is not crucial.

• Physical limits  The length of polygon legs is of no importance according to the interview. With dynamic operators smooth turns are found.

• Safety margin  A safety margin is added by extending the range of the threats.

Some factors in table 1.1 has not been taken into consideration in the simulations. They are: enemy aircraft, weather situation, time restrictions, altitude, logic waypoints, speed and acceleration. Most of them demand a treatment of a dynamic threat scenario and a different threat representation, and some ideas of how to deal with that is presented in chapter 9, Future work.

Contrary to Random restart Hill-climbing and Simulated annealing, A*-search is deterministic. That is, knowing the initial data before a replanning performance, the route can be predicted. That is important for the safety, the stringency and the reliability of the system.

The algorithm time of A*-search is in average 0.85 seconds, with the slowest in 2.08 seconds. It is worth mentioning, that in the simulations a lot of data is stored for later presentation in MATLAB. These procedures consumes some computing time as well, so most likely the true algorithm time is lower than denoted here. 0.85 seconds, which can be seen as an average upper bound of algorithm time, is small enough to work in a real time system. However, some effort may be put to fasten up the system further and to avoid high time peaks.

Dynamic A*-search is a promising method that lower the memory complexity of the search algorithm. It is very fast for a low number of threats in the mission zone. For a large number of threats, normal A*-search is equally fast.

Recalling from section 1.3, the aim of this thesis is to find a safe route to the target point taking the requirements in table 1.1 into consideration. The simulations show that the algorithms used and developed in this thesis meet the goal, and serve as a basis for future development of a mission route replanning system.
9 Future work

In this chapter ideas and suggestions of future work concerning mission route planning are presented.

9.1 Dynamic threats

In this thesis, the threats are static during the search. This is not the case in reality: the shape and direction of the threat area ellipses change with the speed and the bearing of the aircraft. This has to do with the range of a missile and its speed.

![Figure 9.1 Risk areas are dynamic, depending on the aircraft's speed and bearing. However, the impact range is static (described by circles).](image)

There are two technical terms that are used when describing the threat area: impact range and launch zone. The static impact range is the actual range of a missile, described by the circles in figure 9.1. The edge of the launch zone, is the line at which a missile must be fired to hit the aircraft at the edge of the impact zone. The launch zone is dependent on the current speed and bearing of the aircraft (described by dashed lines in figure 9.1).

In a mission route search, the launch zones must be avoided for a safe flight. Therefore, the threats must be regarded as dynamic. That leads to the threat scenario looking different depending on the aircraft bearing and speed, in each step of the A*-search. The idea of putting a risk value
to each node in the map data base before the replanning can no longer be applied. In each step, the risk picture must be updated. On top of the launching zone ellipses come the uncertainties, safety margin and in some cases an extra margin (depending the desirable amount of waypoints). This consumes a lot of computing time, so fast and simple ways of representing threats and calculating the threat cost of a step must be found.

9.2 Mission synchronization and speed

In the same way as fuel is calculated in each step of the A*-search, the consumed time can be regarded. A*-search will then find the optimal route, that takes the aircraft to the target point in time according to the mission time window. There are two ways of changing time-to-arrival: changing speed, or take a route of a different length (that may be riskier). Recalling from section 9.1, speed affects the threat picture (the threat ellipses). Thus, speed is strongly connected to the risk cost of a mission route. Moreover it is connected to the fuel consumption, and dealing with speed as a variable raises the complexity of the replanning situation.

9.3 Flying over threats

It is assumed that the aircraft can be at different altitudes, for example when flying over a mountain. In such cases, when the aircraft increases its altitude, the fuel consumption rises and the risk level of the mission will increase. Even though flying over a threat may sometimes be the optimal option of a route, it is not dealt with in this report.

When making the decision to pass over a threat, the point in which the increasing of altitude is to begin must be determined. It is dependent of the distance to the threat, the speed and the weight of the aircraft. As the range of the threat increases at higher altitudes, this must also be taken into consideration (see figure 9.2).

One approach is to apply A*-search on a 3-dimensional set of states. In that case, it is necessary to find a way to quantify the space in such way that the memory complexity will not be to large. Dynamic operators or a uniform mesh with a few possible altitudes may be two ways to represent a 3-dimensional set of states.

9.4 Physical limits

A*-search should take all physical limits into consideration, for example maximal turn, maximal increasing of altitude and so on. These are factors depending on the weight of the aircraft, including its payload. In case of an emergency situation, the cargo may be dropped so sharper turns can be performed. This is something that the mission replanning system shall have as a possible alternative, if for example the risk level rises very fast and high, or a new threat appears close to the aircraft.

Normally the physical attributes are limited by the MLL (Maximal Load Level) of the aircraft. A*-search with dynamic operators can pay regard to the MLL in such way that the turning angle is dependent on the state of the aircraft. In this thesis, dynamic A*-search pay attention to turning limits successfully, and it should not be any problem to introduce other physical requirements.

9.5 Risk threshold

In this thesis no consideration has been taken to when to replan a mission route. Flying in a mission zone, there is always a risk connected to the mission route, and the pilot does not want the system to replan the route every second.

One way to get around this, is to avoid a replanning as long as the risk value in any point of the route not exceeds some threshold value. The
Improvement of node reduction

replanning system must calculate the risk value continuously in the
background, and when the threshold is exceeded the pilot should be
warned and an alternative route presented.

9.6 Improvement of node reduction

The node-reduction path refinement is an important step when finding
suitable mission routes. In this thesis that algorithm is not optimized, and
modifications would surely improve its performances. For example,
instead of searching from some node p towards the goal node N when
reducing the A* solution path, it could do the opposite. I.e let it search
from node N towards node 1, and repeat this pattern until a route is found
(compare to section 4.1).

By reversing the algorithm, other node-reduced routes would be
presented which in some cases would be better. The best would be to
apply node-reduction path refinement with both directions, and choose
the route that gives the best result according to some evaluation function.

A Simulation data

A.1 Numerical data

The algorithms evaluated are: A*-search (A*), Greedy search (Gr),
Hill climb search (Hi), Random restart hill-climb search (Ra) and
Simulated annealing (Si). Average (avg.) and median values (med.) are
calculated in order to put together the results.

A.1.1 Evaluation function

Evaluation values of the search methods are calculated according to
the evaluation function:

\[
f = 100*\text{threat value} + 40*\text{terrain value} + \text{distance of step} + \text{SLD(goal node)}
\]
### A.1.2 Remaining fuel

<table>
<thead>
<tr>
<th></th>
<th>A*</th>
<th>Gr</th>
<th>Hi</th>
<th>Ra 1</th>
<th>Ra 2</th>
<th>Si 1</th>
<th>Si 2</th>
<th>Best</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>43</td>
<td>42</td>
<td>36</td>
<td>15</td>
<td>24</td>
<td>34</td>
<td>39</td>
<td>A*</td>
<td>Ra</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>17</td>
<td>23</td>
<td>24</td>
<td>17</td>
<td>20</td>
<td>18</td>
<td>A*/Ra</td>
<td>Gr/Ra</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>40</td>
<td>42</td>
<td>15</td>
<td>42</td>
<td>38</td>
<td>38</td>
<td>Hi/Ra</td>
<td>Ra</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>12</td>
<td>28</td>
<td>17</td>
<td>22</td>
<td>28</td>
<td>32</td>
<td>Si</td>
<td>Gr</td>
</tr>
<tr>
<td>5</td>
<td>22</td>
<td>18</td>
<td>34</td>
<td>24</td>
<td>5</td>
<td>22</td>
<td>22</td>
<td>Hi</td>
<td>Ra</td>
</tr>
<tr>
<td>6</td>
<td>45</td>
<td>43</td>
<td>35</td>
<td>34</td>
<td>35</td>
<td>37</td>
<td>A*</td>
<td>Ra</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>18</td>
<td>26</td>
<td>17</td>
<td>27</td>
<td>28</td>
<td>26</td>
<td>Si</td>
<td>A*</td>
</tr>
<tr>
<td>8</td>
<td>19</td>
<td>10</td>
<td>39</td>
<td>10</td>
<td>14</td>
<td>28</td>
<td>31</td>
<td>Hi</td>
<td>Ra</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>14</td>
<td>37</td>
<td>19</td>
<td>15</td>
<td>31</td>
<td>33</td>
<td>Hi</td>
<td>Ra</td>
</tr>
<tr>
<td>10</td>
<td>28</td>
<td>12</td>
<td>21</td>
<td>24</td>
<td>27</td>
<td>4</td>
<td>26</td>
<td>A*</td>
<td>Si</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>35</td>
<td>36</td>
<td>15</td>
<td>12</td>
<td>33</td>
<td>37</td>
<td>Si</td>
<td>Ra</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>14</td>
<td>25</td>
<td>8</td>
<td>29</td>
<td>26</td>
<td>12</td>
<td>Ra</td>
<td>Ra</td>
</tr>
<tr>
<td>13</td>
<td>34</td>
<td>20</td>
<td>30</td>
<td>19</td>
<td>30</td>
<td>32</td>
<td>27</td>
<td>A*</td>
<td>Ra</td>
</tr>
<tr>
<td>avg</td>
<td>22</td>
<td>23</td>
<td>32</td>
<td>18</td>
<td>28</td>
<td>28</td>
<td>29</td>
<td>Hi/Ra</td>
<td>Ra</td>
</tr>
<tr>
<td>med</td>
<td>22</td>
<td>18</td>
<td>34</td>
<td>18</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>Hi/Ra</td>
<td>Ra</td>
</tr>
</tbody>
</table>

*Table A.2* Remaining fuel at target point.

The numbers in table A.2 are the relation between the remaining fuel and the fuel that is demanded to reach the target point (in percentages). Four times Random restart hill-climb search chooses routes that had too high fuel consumption (encircled in table A.2).

### A.1.3 Algorithm time

<table>
<thead>
<tr>
<th></th>
<th>A*</th>
<th>Gr</th>
<th>Hi</th>
<th>Ra 1</th>
<th>Ra 2</th>
<th>Si 1</th>
<th>Si 2</th>
<th>Best</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.28</td>
<td>0.06</td>
<td>0.76</td>
<td>10.16</td>
<td>5.86</td>
<td>4.56</td>
<td>4.42</td>
<td>Gr</td>
<td>Ra</td>
</tr>
<tr>
<td>2</td>
<td>0.58</td>
<td>0.16</td>
<td>1.67</td>
<td>5.22</td>
<td>7.09</td>
<td>5.85</td>
<td>5.84</td>
<td>Gr</td>
<td>Ra</td>
</tr>
<tr>
<td>3</td>
<td>0.83</td>
<td>0.16</td>
<td>1.13</td>
<td>6.92</td>
<td>10.76</td>
<td>5.74</td>
<td>5.07</td>
<td>Gr</td>
<td>Ra</td>
</tr>
<tr>
<td>4</td>
<td>0.64</td>
<td>0.24</td>
<td>1.17</td>
<td>12.21</td>
<td>7.08</td>
<td>5.32</td>
<td>5.35</td>
<td>Gr</td>
<td>Ra</td>
</tr>
<tr>
<td>5</td>
<td>0.78</td>
<td>0.25</td>
<td>1.22</td>
<td>8.48</td>
<td>9.49</td>
<td>5.62</td>
<td>5.67</td>
<td>Gr</td>
<td>Ra</td>
</tr>
<tr>
<td>6</td>
<td>0.27</td>
<td>0.08</td>
<td>0.41</td>
<td>5.98</td>
<td>5.45</td>
<td>4.81</td>
<td>4.66</td>
<td>Gr</td>
<td>Ra</td>
</tr>
<tr>
<td>7</td>
<td>0.58</td>
<td>0.17</td>
<td>1.01</td>
<td>8.28</td>
<td>8.43</td>
<td>6.68</td>
<td>6.25</td>
<td>Gr</td>
<td>Ra</td>
</tr>
<tr>
<td>8</td>
<td>0.97</td>
<td>0.22</td>
<td>1.45</td>
<td>7.84</td>
<td>10.72</td>
<td>5.21</td>
<td>5.75</td>
<td>Gr</td>
<td>Ra</td>
</tr>
<tr>
<td>9</td>
<td>0.99</td>
<td>0.30</td>
<td>1.40</td>
<td>6.43</td>
<td>10.38</td>
<td>5.42</td>
<td>5.59</td>
<td>Gr</td>
<td>Ra</td>
</tr>
<tr>
<td>10</td>
<td>2.08</td>
<td>0.17</td>
<td>2.84</td>
<td>8.94</td>
<td>8.45</td>
<td>8.02</td>
<td>6.83</td>
<td>Gr</td>
<td>Ra</td>
</tr>
<tr>
<td>11</td>
<td>0.53</td>
<td>0.08</td>
<td>0.93</td>
<td>10.03</td>
<td>9.85</td>
<td>4.95</td>
<td>5.34</td>
<td>Gr</td>
<td>Ra</td>
</tr>
<tr>
<td>12</td>
<td>2.04</td>
<td>0.26</td>
<td>2.66</td>
<td>10.14</td>
<td>9.94</td>
<td>6.28</td>
<td>7.41</td>
<td>Gr</td>
<td>Ra</td>
</tr>
<tr>
<td>13</td>
<td>0.53</td>
<td>0.10</td>
<td>1.05</td>
<td>6.99</td>
<td>8.9</td>
<td>5.72</td>
<td>5.41</td>
<td>Gr</td>
<td>Ra</td>
</tr>
</tbody>
</table>

| avg | 0.85  | 0.17  | 1.36 | 8.34 | 5.68 | 5.61 | 5.61 | Gr   | Ra    |
| med | 0.64  | 0.17  | 1.17 | 8.47 | 5.61 | 5.61 | 5.61 | Gr   | Ra    |

*Table A.3* Algorithm times for the search methods (in seconds).

In table A.3 the algorithm times of the different search methods are shown.
### A.1.4 Length of route

<table>
<thead>
<tr>
<th></th>
<th>A*</th>
<th>Gr</th>
<th>Hi</th>
<th>Ra 1</th>
<th>Ra 2</th>
<th>Si 1</th>
<th>Si 2</th>
<th>Best</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>11</td>
<td>15</td>
<td>56</td>
<td>39</td>
<td>20</td>
<td>11</td>
<td>A*</td>
<td>Ra</td>
</tr>
<tr>
<td>2</td>
<td>29</td>
<td>29</td>
<td>28</td>
<td>26</td>
<td>39</td>
<td>33</td>
<td>36</td>
<td>Ra</td>
<td>Ra</td>
</tr>
<tr>
<td>3</td>
<td>73</td>
<td>3</td>
<td>3</td>
<td>36</td>
<td>4</td>
<td>16</td>
<td>10</td>
<td>Ra</td>
<td>Hi/Ra</td>
</tr>
<tr>
<td>4</td>
<td>37</td>
<td>37</td>
<td>28</td>
<td>49</td>
<td>39</td>
<td>30</td>
<td>21</td>
<td>Si</td>
<td>Ra</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>16</td>
<td>5</td>
<td>20</td>
<td>50</td>
<td>24</td>
<td>23</td>
<td>Hi</td>
<td>Ra</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>7</td>
<td>17</td>
<td>17</td>
<td>15</td>
<td>11</td>
<td>A*</td>
<td>Ra</td>
</tr>
<tr>
<td>5</td>
<td>26</td>
<td>10</td>
<td>15</td>
<td>30</td>
<td>13</td>
<td>12</td>
<td>15</td>
<td>Gr</td>
<td>Ra</td>
</tr>
<tr>
<td>8</td>
<td>46</td>
<td>46</td>
<td>8</td>
<td>79</td>
<td>54</td>
<td>28</td>
<td>21</td>
<td>Hi</td>
<td>Ra</td>
</tr>
<tr>
<td>9</td>
<td>62</td>
<td>52</td>
<td>8</td>
<td>107</td>
<td>45</td>
<td>19</td>
<td>14</td>
<td>Hi</td>
<td>Ra</td>
</tr>
<tr>
<td>10</td>
<td>37</td>
<td>43</td>
<td>41</td>
<td>34</td>
<td>33</td>
<td>71</td>
<td>34</td>
<td>Ra</td>
<td>Si</td>
</tr>
<tr>
<td>11</td>
<td>51</td>
<td>5</td>
<td>8</td>
<td>44</td>
<td>73</td>
<td>15</td>
<td>7</td>
<td>Gr</td>
<td>Ra</td>
</tr>
<tr>
<td>12</td>
<td>39</td>
<td>20</td>
<td>30</td>
<td>58</td>
<td>22</td>
<td>26</td>
<td>50</td>
<td>Gr</td>
<td>Ra</td>
</tr>
<tr>
<td>13</td>
<td>6</td>
<td>21</td>
<td>9</td>
<td>26</td>
<td>10</td>
<td>6</td>
<td>16</td>
<td>A*/Si</td>
<td>Ra</td>
</tr>
<tr>
<td>avg</td>
<td>34</td>
<td>23</td>
<td>15</td>
<td>41</td>
<td>22</td>
<td>Hi</td>
<td>Ra</td>
<td></td>
<td></td>
</tr>
<tr>
<td>med</td>
<td>37</td>
<td>20</td>
<td>9</td>
<td>39</td>
<td>20</td>
<td>Hi</td>
<td>Ra</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table A.4  Relative length of the mission routes.*

In table A.4 the relative length of the mission routes are shown. It is calculated by the relation between the route found by the search method and the straight line distance between the start and the goal node (in percentages).

### A.1.5 Crossing threat area

<table>
<thead>
<tr>
<th></th>
<th>A*</th>
<th>Gr</th>
<th>Hi</th>
<th>Ra 1</th>
<th>Ra 2</th>
<th>Si 1</th>
<th>Si 2</th>
<th>Best</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Hi</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>1</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>Hi</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>A*</td>
<td>Gr</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>8</td>
<td>Si</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>31</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>A*/Ra</td>
<td>Gr</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Hi</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Hi</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Hi</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>9</td>
<td>A*</td>
<td>Si</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>A*</td>
<td>Gr</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>Si</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>12</td>
<td>1</td>
<td>Si</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Hi</td>
<td></td>
</tr>
<tr>
<td>avg</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>A*</td>
<td>Gr/Hi</td>
<td></td>
<td></td>
</tr>
<tr>
<td>med</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>Hi</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table A.5  Part of the route that cuts a threat area (in percentages).*

The numbers in table A.5 is the part of the mission route that crosses a threat area (in percentages).
A.1.6 Highest terrain peak

<table>
<thead>
<tr>
<th></th>
<th>A*</th>
<th>Gr</th>
<th>Hi</th>
<th>Ra 1</th>
<th>Ra 2</th>
<th>Si 1</th>
<th>Si 2</th>
<th>Best</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>9</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Gr</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>9</td>
<td>29</td>
<td>99</td>
<td>0</td>
<td>30</td>
<td>99</td>
<td>Ra</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>25</td>
<td>31</td>
<td>95</td>
<td>28</td>
<td>27</td>
<td>A*</td>
<td>Ra</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>10</td>
<td>64</td>
<td>0</td>
<td>0</td>
<td>84</td>
<td>86</td>
<td>Si</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>105</td>
<td>20</td>
<td>46</td>
<td>5</td>
<td>16</td>
<td>14</td>
<td>Ra</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>A*</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>19</td>
<td>19</td>
<td>22</td>
<td>Si</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>80</td>
<td>0</td>
<td>0</td>
<td>85</td>
<td>80</td>
<td>Si</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>11</td>
<td>87</td>
<td>81</td>
<td>102</td>
<td>89</td>
<td>89</td>
<td>A*</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>43</td>
<td>58</td>
<td>72</td>
<td>101</td>
<td>45</td>
<td>67</td>
<td>45</td>
<td>A*</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>17</td>
<td>13</td>
<td>57</td>
<td>74</td>
<td>11</td>
<td>17</td>
<td>A*</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>36</td>
<td>36</td>
<td>72</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>61</td>
<td>Hi</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>10</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td>10</td>
<td>24</td>
<td>6</td>
<td>Gr/Ra</td>
<td></td>
</tr>
<tr>
<td>avg.</td>
<td>12</td>
<td>24</td>
<td>42</td>
<td>33</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>A*</td>
<td></td>
</tr>
<tr>
<td>med.</td>
<td>9</td>
<td>11</td>
<td>29</td>
<td>15</td>
<td>28</td>
<td>28</td>
<td>28</td>
<td>Gr</td>
<td></td>
</tr>
</tbody>
</table>

Table A.6 The highest terrain peak that the mission route crosses.

In the simulations the terrain had no connection to real physical values. The terrain value ranged from 0 to approximately 200. In table A.6, the highest terrain peak of the mission routes are found.

A.2 A*-search mission route suggestions

The final result of mission route planning is presented for the 13 scenarios. Each full line corresponds to normal A*-search, i.e. A*-search with static successors applied on a quantified map grid. Each dashed line corresponds to dynamic A*-search.

Sometimes very strange routes are presented (for example in scenario 9). That is a result of the limitations of the turning angle. The bearing at the first waypoint is always 90 degrees east, and no predefined approach bearing is set.

A.2.1 Scenario 1

Figure 1.1 Static A*-search (full line) and dynamic A*-search (dashed line).
A.2.2 Scenario 2

Figure 1.2 Static A*-search (full line) and dynamic A*-search (dashed line).

A.2.3 Scenario 3

Figure 1.3 Static A*-search (full line) and dynamic A*-search (dashed line).

A.2.4 Scenario 4

Figure 1.4 Static A*-search (full line) and dynamic A*-search (dashed line).

A.2.5 Scenario 5

Figure 1.5 Static A*-search (full line) and dynamic A*-search (dashed line).
A.2.6 Scenario 6

Figure 1.6 Static A*-search (full line) and dynamic A*-search (dashed line).

A.2.7 Scenario 7

Figure 1.7 Static A*-search (full line) and dynamic A*-search (dashed line).

A.2.8 Scenario 8

Figure 1.8 Static A*-search (full line) and dynamic A*-search (dashed line).

A.2.9 Scenario 9

Figure 1.9 Static A*-search (full line) and dynamic A*-search (dashed line).
A.2.10 Scenario 10

Figure 1.10 Static A*-search (full line) and dynamic A*-search (dashed line).

A.2.11 Scenario 11

Figure 1.11 Static A*-search (full line) and dynamic A*-search (dashed line).

A.2.12 Scenario 12

Figure 1.12 Static A*-search (full line) and dynamic A*-search (dashed line).

A.2.13 Scenario 13

Figure 1.13 Static A*-search (full line) and dynamic A*-search (dashed line).
References

Reasoning about plans
ISBN 1-55860-137-6
Morgan Kaufmann Publishers Inc. San Mateo, USA 1991

[And94] Magnus Andersson & Urban Blomberg
Banplanering: en studie med rätlinjiga banor bland elliptiska hinder
Master Thesis report LiTH-ISY-EX-1523
Department of Electrical Engineering
Linköping University, Linköping, Sweden 1994

Multi-sensor fusion: fundamentals and applications with software
Prentice Hall PTR, New Jersey, USA 1998

[Caw98] Alison Cawsey
The essence of artificial intelligence
Prentice Hall Europe, Hampshire, Great Britain 1998

[Dur97] Edmund H. Durfee, Patrick G. Kenny & Karl C. Kluge
Integrated Premission Planning and Execution for Unmanned Ground Vehicles
Article DAAE07-92-CR012
EECS Department at ACM
University of Michigan, Michigan, USA 1997

[Fab99] Patrick Fabiani & Jean-Claude Latombe
Planning and scheduling: Dealing with geometric constraints in game-theoretic planning, p. 942-947
Article DAAH04-96-1-007
Department of Computer Science
Stanford university, Stanford 1999

[Fer98] Niklas Ferm
Identity Fusion and Classification of Aircraft in a Multisensor Environment
Master Thesis report LiTH-ISY-EX-1985
Department of Electrical Engineering
Linköping University, Linköping, Sweden 1998

[Jen97] Thomas Jensen & Mathias Karlsson
Sensor Management in a Complex Multisensor Environment
Master Thesis report LiTH-ISY-EX-1831
Department of Electrical Engineering
Linköping University, Linköping, Sweden 1997
[Kle98] Michel R. Klein & Leif B. Methlie
Knowledge-based Decision Support Systems
ISBN 0-471-95295-8
John Wiley & Sons Ltd, West Sussex, England 1998

[Lin99] Anna Lindahl
Omplanering av uppdrag i realtid: beslutstidsmetoder för JAS 39 Gripen
Master Thesis report LiTH-IDA-Ex-99/N
Department of Computer and Information Science
Linköping University, Linköping, Sweden 1999

[Rus95] Stuart J. Russell & Peter Norvig
Artificial Intelligence: a modern approach
Prentice-Hall Inc, New Jersey, USA 1995

[Sch98] Henrik Schinzel
Automated Fighter Pilot Decision Making
Master Thesis report nr 98-x-074 (DSV)
School of Computer Science and Technology
Stockholm University, Stockholm 1998

[Ska96] Jan Skansholm
C++ direkt
ISBN 91-44-47931-X
Studentlitteratur, Lund, Sweden 1996

[Sno99] Magnús Snorrason, Jeff Norris & Paul Backes
Vision based obstacle detection and path planning for planetary rovers
Article No. NASA-98192
Jet Propulsion Laboratory at NASA
Pasadena, USA 1999