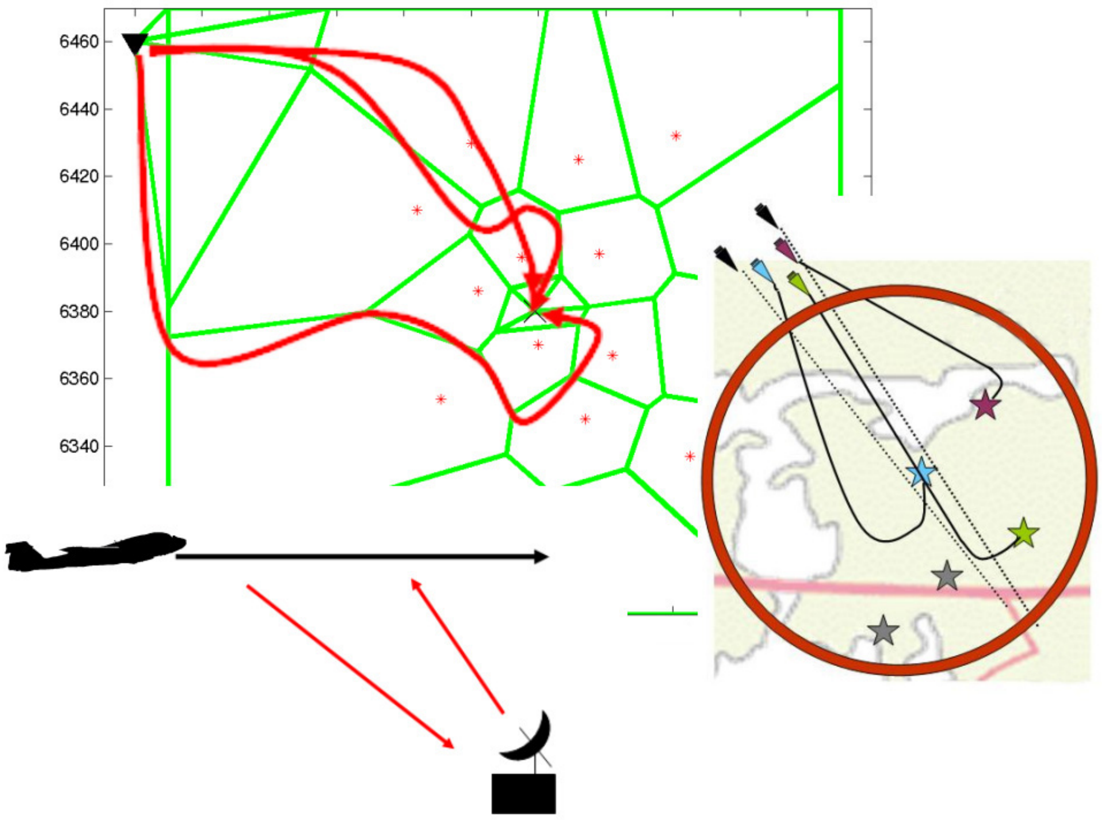
FOI-R--1382--SE
October 2004
1650-1942

Scientific report

Maja Winstrand

Mission Planning and Control of Multiple UAVs



System Technology Division
SE-172 90 STOCKHOLM
Sweden

Swedish Defence Research Agency
System Technology Division
SE-172 90 STOCKHOLM
Sweden

FOI-R--1382--SE
October 2004
1650-1942

Scientific report

Maja Winstrand

Mission Planning and Control of Multiple UAVs

Issuing organization Swedish Defence Research Agency System Technology Division SE-172 90 STOCKHOLM Sweden	Report number, ISRN FOI-R--1382--SE	Report type Scientific report
	Research area code Combat	
	Month year October 2004	Project no. E6003
	Customers code Commissioned Research	
	Sub area code Weapons and Protection	
Author/s (editor/s) Maja Winstrand	Project manager Peter Alvå	
	Approved by Monica Dahlén	
	Sponsoring agency Swedish Armed Forces	
	Scientifically and technically responsible Xiaoming Hu	

Report title

Mission Planning and Control of Multiple UAVs

Abstract

In this master thesis two different multi UAV problems are considered. The first one concerns planning and execution of a suppression of enemy air defence (SEAD) mission. We build upon earlier work by adding the important element of target assignment to the path planning, i.e. deciding what threats to fly around and what threats to engage. This is done by extending the state space to include the number of available High speed Anti-Radiation Missiles (HARMs). The possibility of considering different enemy radar ranges is also accommodated by adding extra edges to the initial Voronoi graph discretization. The planning output can further more be used as online decision support, since it contains estimated mission success probabilities for different position and armament combinations. Finally, the issue of timing the engagement of a single enemy surface to air missile (SAM) target is studied using a game theoretic dynamic programming approach.
The second multi UAV problem considered is that of trajectory planning and target assignment on a much smaller scale, such as the final phase of a strike against multiple moving ground targets. To take advantage of the continuously arriving sensor data and respect the kinematic constraints, such as the vehicle minimum turning radius, we propose an algorithm that iteratively re-plans target assignments and trajectories. Finally, the algorithm accommodates prioritization of different target categories as well as a scheme for achieving scouting, concurrent strike and battle damage assessment (BDA).

Keywords

Cooperation, multi-agent coordination, path planning, trajectory generation, UAV

Further bibliographic information	Language English
ISSN 1650-1942	Pages 52
Distribution By sendlist	Price Acc. to pricelist Security classification Unclassified

Utgivare Totalförsvarets forskningsinstitut Avdelningen för Systemteknik SE-172 90 STOCKHOLM Sweden	Rapportnummer, ISRN FOI-R--1382--SE	Klassificering Vetenskaplig rapport
	Forskningsområde Bekämpning	
	Månad, år Oktober 2004	Projektnummer E6003
	Verksamhetsgren Uppdragsfinansierad verksamhet	
	Delområde VVS med styrda vapen	
Författare/redaktör Maja Winstrand	Projektledare Peter Alvå	
	Godkänd av Monica Dahlén	
	Uppdragsgivare/kundbeteckning FM	
	Tekniskt och/eller vetenskapligt ansvarig Xiaoming Hu	

Rapportens titel

Uppdragsplanering och styrning av flera UAVer

Sammanfattning

Detta examensarbete behandlar två olika problem rörande en grupp UAV:er. Det första är ett så kallat SEAD-uppdrag (Suppression of Enemy Air Defence) där gruppen skall ta sig igenom ett område bevakat av fientligt luftvärn. Vi bygger på tidigare metoder genom att använda en Voronoigraf som underlag i banplaneringen. Vi utökar tillståndsrummet genom att lägga till en dimension för antalet signalsökande robotar och får på så sätt en kombinerad rutt- och målvalsplanering. Genom att lägga till ytterligare kanter i grafen tas även de olika radarräckvidderna med i vägvalet. Utdata från planeringsverktyget kan även användas som underlag för beslut under uppdragets gång då uppskattade genomförbarhetssannolikheter finns tillgängliga för olika kombinationer av positioner och resurser i form av signalsökande robotar. Slutligen studerar vi även vilka avfyrningstider som är optimala under anflygning mot en enskild luftvärnsanläggning. En spelteoretisk ansats tillsammans med dynamisk programmering används för att fånga beteendet hos en intelligent motståndare.
I den andra delen av examensarbetet behandlas ruttplanering och målval på en mycket mindre skala, som t.ex. slutfasen på en attack mot flera rörliga markmål. I en sådan situation måste både den ständiga uppdateringen av sensordata och de kinematiska begränsningarna i form av minimal svängradie hos farkosten tas med i beräkningarna. För att uppnå detta föreslår vi en algoritm som iterativt omplannerar målval och banval för alla farkoster baserat på den senaste sensorinformationen. Olika måltyper kan ges olika prioritet och spaning, samtidig träff och verkansvärdering kan hanteras.

Nyckelord

Samverkan, banplanering, obemannade flygplan, UAV

Övriga bibliografiska uppgifter	Språk Engelska
ISSN 1650-1942	Antal sidor 52
Distribution Enligt missiv	Pris Enligt prislista Sekretess Öppen

Contents

Acknowledgements

This work was conducted at the Swedish Defence Research Agency (FOI), Department of Systems Technology, from February to September 2004, as part of my MSc degree in Engineering Physics, majoring in Optimization and Systems theory, at the Royal Institute of Technology, Stockholm.

My thanks go to my examiner, Prof. Xiaoming Hu, for his help with establishing contact with the Swedish Defence Research Agency. I also owe much gratitude to my supervisor, Dr. Petter Ögren, without whom this work would not have been possible. Always cheerful and with a good share of humour he has guided me through the work.

Finally I want to thank Magnus and Fredrik for being my invaluable company during these months and Anna, Maja and Linda for showing me what it is to write a master thesis.

1. Introduction

The overall goal for this research project is to develop methods and techniques for co-operative and autonomous control of systems containing several unmanned air vehicles (UAVs).

The main focus is on so-called SEAD (Suppression of Enemy Air Defence) missions, where the goal is to temporarily or permanently neutralize the air defence either by forcing it to shut down or by destroying it. To accomplish this the UAVs are equipped with so called HARMs (High Speed Anti-Radiation Missiles). The fact that this is a high risk mission in a partly unknown environment makes the potential benefits of an unmanned system substantial. The second focus is on a scenario where multiple moving targets are attacked by multiple agents, requiring target assignment of prioritized targets and path planning in real time, denoted the Multiple Target Engagement Problem.

The two scenarios can be thought of as successive parts of a single mission, the multiple targets-scenario taking place at the target area of the SEAD mission.

1.1 Scenarios and Subproblems

To study the techniques developed, a scenario is defined. This scenario includes four UAVs clearing a path through an area covered by an air defence network. All the UAVs are heading for the target area, located in or beyond the air defence area. This first part of the problem is a SEAD problem. In the target area a number of moving targets, e.g. tanks, are located. A fifth, so far passive UAV, now drops a number of guided missiles to hit the targets in prioritized order. Alternatively, the UAVs could be equipped with extra air-to-ground ordnance, and the target engagement is performed by the five UAVs together. This second part of the main problem is the Multiple Target Engagement Problem.

The problem is to find a good strategy for the five UAVs carrying out the mission, including a path to follow, firing instructions and handling of unexpected events, that yields a high probability to survive. To fulfil this aim, the vehicles are required to cooperate in locating and reacting to events in the environment.

The approach used is to divide the mission into a sequence of subproblems, each forming a problem that can be solved with techniques from the fields of optimization and control. This division is by no means unique or straightforward and some care needs to be taken in order not to lose too much of the original problem structure. The main parts in this division are timing, path planning, target assignment and target engagement.

Although the individual subsystems will be analysed with theoretical tools it seems that the only available overall performance evaluation method is simulation. Thus there is a strong need for a commonly accepted benchmark problem.

Below, the subproblems are described briefly. More substantial descriptions are given in chapters 3 to 5, also including the proposed solutions to each subproblem.

Timing Tactics Against Single SAM Opponent This subproblem handles the question of timing. When a UAV approaches one of the components in the air defence, a so called SAM (Surface-to-Air Missile) site, a gaming situation occurs, where the objective of the UAV is to pass the SAM site unhurt, by suppressing or destroying it, while the SAM site strives to prevent the UAV from passing and to minimize the risk of being hit. There might also be other objectives, such as minimizing the number of enemies in action or minimizing the monetary cost of the operation. The goal here is to find the optimal moment for the UAV to fire the available HARMs.

Path Planning and Target Assignment/Resource Allocation The Path Planning problem is the problem of assigning a feasible path for each of the four UAVs, such that the fifth aircraft reaches the target area as safely as possible, while also minimizing the risk for each of the SEAD aircraft. The solution should handle so called pop-up threats as well as SAM sites of which the location is known in advance.

Since the number of HARMs available in the mission is limited, a priority should be given to each threat, so that the SAM sites that contribute most to the overall threat are attacked. The goal is to assign HARMs to a sufficient number of SAM sites so that the overall threat is acceptable and the mission succeeds with a high probability, while still saving a number of HARMs for possible unknown pop up-threats.

Multiple Target Engagement Once in the target area there is the problem of target engagement. There might be several targets to handle, stationary as well as moving. A method for finding a good strategy for multiple UAVs engaging multiple moving targets is sought for.

1.1.1 Contributions and Previous Work The SEAD problem is a relatively new field of interest to mathematical researchers. However, the path planning part of the problem has a long history. Finding the shortest path from one point to another and between various geometrical figures was a task that thrilled even Greek thinkers some hundred years BC. Path planning as a matter of control theory is more complicated and a thorough treatment of path planning for autonomous systems can be found in [11]. When it comes to path planning in the presence of stationary obstacles, as is the case in a SEAD problem, a common approach is the use of Voronoi diagrams to reduce the number of possible paths. This is done in e.g. [4], where a Voronoi diagram is placed around the SAM sites, yielding a path that is guaranteed to pass as far away as possible from the nearest threat sources. In this thesis the concept is taken further, using an extended Voronoi grid. New edges are added at the borders of the SAM sites' ranges, taking into account that the maximization of the distance to the SAM sites not necessarily minimizes the threat. Furthermore, the grid is expanded to a three dimensional multi-layer grid, capturing not only the position of the UAV in space, but also the engagement with the different SAM sites. This approach includes the question of target assignment into the path planning. Finally, a cost definition is proposed to make the natural objective function of success probability carry over to the expanded Voronoi framework.

The solution to the problem of timing tactics in the SEAD mission, as proposed in this thesis, using dynamic programming to solve a discrete dynamic zero-sum game, is, to the best of our knowledge, not previously reported in the literature.

The Multiple Target Engagement Problem, discussed in the thesis' last part, is inspired by the situation described in [1]. The problem formulation is well adapted to methods in linear programming and optimal control. The solution proposes a combination of Sussmann and Tang's shortest path algorithm, [2], and a P-regulator and ends up with a handy scheme for finding the proper UAV behaviour.

1.1.2 What is Success? To evaluate different approaches it is important to know what properties are desired in a proposed solution. Only when these properties are well defined it will be possible to talk about good or bad methods. The *success rate* is a helpful measure when comparing different methods. This quota reveals in how many cases the mission will be carried out successfully. In this work the success rate will be based on the probabilities of a mission failure, and henceforth *success rate* will denote expected success rate.

A minimum requirement for accepting a new method is that the success rate exceeds the threshold levels specified for the mission in question. A balance between the simplicity of the method and it's success rate then has to be done. Since all methods are based on abstractions of real life, the true success rate when using the method will of course differ from the theoretical one.

The event of mission failure may denote a variety of outcomes, from making insufficient damage to one of the targets to loosing the whole UAV fleet. Through the first chapters of this thesis (chapters 3 and 4, considering the SEAD part) a mission will be considered successful if all the UAVs reach the target area. In chapter 5 the final part of the mission is studied, where the target area is reached and multiple moving targets are engaged. Here mission success is defined as all targets being hit.

1.2 Background

The systems treated in this thesis are simplified models of real systems in operation at present or in the near future. A short review of the real components of a SEAD mission is given below.

1.2.1 UAVs The history of unmanned aerial vehicles, UAVs, is longer than is usually assumed, starting with primitive vehicles in the early 20th century [20]. The UAVs came into regular use during the Second World War and UAVs have since then been used and developed by many nations [19]. The successful American use of reconnaissance UAVs during the Gulf War has become a widely spread example when it comes to stating the advantages to be gained by using UAVs.

Today the UAVs are mainly used for intelligence, surveillance and reconnaissance (ISR) missions. There is however a wish to extend the field of application to combat missions. UAVs intended for this type of missions, carrying offensive bombs or missiles, forms a subgroup of the UAV family called uninhabited combat air vehicles, UCAVs. An example of a UCAV test bed that has been used in some test flights is the SHARC developed by SAAB, shown in picture 1.1 below.

The potential advantages of UCAVs over ordinary, manned combat aircraft are many. The most obvious is the possibility to perform high-risk missions without endangering the life of a pilot, but there are also improvements in performance and costs. Since pilot requirements, such as cockpit controls and interface between human and machine, oxygen systems and pressure control are unnecessary, the construction of a UCAV can be cheaper than for manned aircraft. The total life time cost of the UCAV will also be lower, since training of the operators can be done in simulators, letting the UCAVs fly less frequently, rendering much lower maintenance costs. Furthermore, the slimmed construction implies lower weight, which saves considerable amounts of fuel. In total, the life cycle cost is expected to be reduced by 50-80%, [8]. The performance of an unmanned system can be made more extreme, since there are no pilot-induced limits on flight time or high, 10G+, turns. The aircraft can also be made smaller, which lowers the radar cross section and increases the survivability, and without a pilot, operations can be executed in nuclear, biological or chemical environments [19].

The UCAVs can carry any type of air-to-ground ordnance, or a variety of missile types. In this thesis the missiles used in the SEAD part are so called High Speed Anti-Radiation Missiles, HARMs, or Air Launched Anti-Radiation Missiles (ALARMs). Both types are equipped with radar receivers which guide the missile towards radar emitting ground-based targets. The HARM heads straight for the target, while the ALARM climbs to a high altitude position above the target. At this high position it deploys a parachute, and descend slowly towards the target, seeking for a signal from the threatening radar site. Once the signal is detected, the parachute is removed and the missile steers towards the target, using only gravity to accelerate [21].

Throughout the thesis the unmanned aircraft will be denoted by the generic term UAV, though it is sometimes clear that the vehicles in view are UCAVs. The missiles fired from the aircraft will be named HARMs, though both of the above mentioned types are applicable.

1.2.2 The Air Defence System Ever since aircraft became an important part of warfare, during the First World War, there has been a need for an effective air defence system. As the aircraft developed, the air defence evolved too, to meet the increased threat from the air. Starting with primitive equipment, such as mounting an m/02 gun on top of a stump, [18], the modern air defence of today is a highly technological, integrated system covering and defending large areas.

The modern air defence system consists of three main parts, viz. sensors, fire unit and command [13]. The sensors are mostly radar or, occasionally, infrared sensors of different ranges. As a threat is detected by the sensors, the information is treated and evaluated, and if found appropriate, the fire unit launches a Surface-to-Air Missile (SAM) towards the threat.

In modern air defence, the sensors in an area are linked so that information about the enemy's position and movements can be communicated throughout the whole air defence system, offering a better foundation for the decisions. In such a system, called integrated air defence system, IADS, the loss of one radar is of less significance to the overall performance.

The sensors are the part of the system that detects the enemy, but it is also the part that reveals the presence of the air defence to the enemy. A site emitting radar signals can be detected by radar warning receivers carried by a hostile aircraft. In other words, looking for others renders yourself visible. This is used by combat aircraft on a SEAD mission, and once detected and shot-at, the sensors must be shut down to avoid a fatal hit. Shutting down the sensors, however, causes any radar guided missiles launched by the fire unit to be lost. To avoid being detected by the

Figure 1.1: The SHARC (Swedish Highly Advanced Research Configuration) developed by SAAB. Courtesy SAAB.

attacking enemy aircraft, the radar sensors can be used in a blinking mode, turning on and off the signals, being visible for about 20 seconds at a time, [10]. Using this tactics in an IADS, sharing the information about the enemy location with sensors located in other areas, makes it possible for the air defence system to stay relatively silent while still keeping track of the enemy with high accuracy. Unguided SAMs may then be fired at the aircraft, using it's estimated position as target.

In early air defence systems the sensors and the fire unit were located close to each other, often mounted on the same vehicle, since the information from the sensors had to be transmitted directly to the fire unit, [21]. In an IADS the fire units use information from multiple sensors, that are not necessarily located in its close vicinity. This is a considerable improvement since the sensors reveal only their own positions by using the radar, while the positions of the fire units are still unknown to the enemy. The performance of the air defence system is then not significantly lowered if one of the sensors is lost.

In this thesis the air defence system is supposed to be a communicating network, but to simplify, the fire units are still assumed to be dependent on their own sensor. The sensor and the fire unit are considered as one unit, located at a single spot, and throughout the thesis referred to as a SAM site.

SAM sites of two ranges are used, SA-8 and SA-6, of 15 and 24 km range, respectively. These low- and low to medium altitude systems were developed in the Soviet Union in the 1960's, the first carrying 6 missiles while the latter is equipped with only 3, [6]. In this thesis the number of available missiles will however not be considered.

1.3 Outline of Thesis

Some mathematical tools used in the thesis are presented in chapter 2 below. In the three following chapters the subproblems of this thesis, as presented in section 1.1 above, are presented together with their proposed solutions. The first two of these, chapters 3 and 4, refer to the first focus of the thesis, the SEAD problem, and finds strategies for the UAVs as they cross the area covered by enemy air defence. The second focus of the thesis, the Multiple Target Engagement Problem, is handled in the third and last of these problem describing chapters, 5. Here a solution is proposed on how to find a proper strategy for multiple UAVs engaging multiple moving targets. Chapter 6 finally states the conclusions to be drawn of this work and proposes some possible future extensions.

2. Mathematical Preliminaries

In this chapter some of the mathematical concepts used in the thesis are presented. Since the emphasis will be on modelling and problem formulation, the rest of the report will be readable without the content below. The proposed solutions mostly refer to this chapter for the mathematical algorithms. However, to understand the proposed solutions in detail, the reader must be familiar with the following mathematical tools.

2.1 Optimal Control and Dynamic Programming

Optimal control concerns systematically finding the best solution to a control problem, according to some stated criterion. The field includes, among other approaches, dynamic programming, the Pontryagin minimum principle and linear quadratic control, of which dynamic programming will be presented in this section. First a standard formulation of an optimal control problem is stated.

2.1.1 The Optimal Control Problem An optimal control problem consists of system dynamics, boundary conditions, control constraints and a cost criterion. The system dynamics, $\dot{x} = f(t, x, u)$ describes the evolution of the state $x = [x_1, x_2 \dots x_n] \in \mathcal{X} \subseteq R^n$ in time depending on the control variable $u = [u_1 \dots u_m] \in \mathcal{U} \subseteq R^m$. The initial and terminal states, $x(t_i)$ and $x(t_f)$, are constrained to some manifolds S_i and S_f. A free initial or terminal state is obtained by letting the manifold equal $\mathcal{X}$.

The cost criterion, $\int_{t_i}^{t_f} \mathcal{L}(t, x(t), u(t))\mathrm{d}t$, is a function of the state at all points from $x(t_i)$ to $x(t_f)$, and gives a weight to states that deviates from what is desired. In the solution of the optimal control problem the cost function is minimized and the state is thus forced to take on values as close as possible to the desired values. It is thus the cost criterion that defines what states are desired.

The cost function often contains one term $\int_{t_i}^{t_f} f_0(t, x(t), u(t))\mathrm{d}t$, defining the wanted trajectories of the state and control, and a second term, $\phi(x(t_f))$, driving the final state towards a certain value.

To conclude, the optimal control problem is formulated as

$$\begin{array}{ll} \min & \phi(x(t_f)) + \int_{t_i}^{t_f} f_0(t, x(t), u(t))\mathrm{d}t \\ \text{subj. to} & \begin{cases} \dot{x}(t) = f(t, x(t), u(t)), \;\; u^*(t) \in \mathcal{U} \\ x(t_i) \in S_i, \\ x(t_f) \in S_f, \end{cases} \end{array}$$

The solution found by dynamic programming or any other approach is a control law $u^*(\cdot)$, that minimizes the cost function.

2.1.2 Dynamic Programming The method of dynamic programming was developed by Richard Bellman in the 1950's and is based on the intuitive, yet profound *principle of optimality*:

> *The restriction of an optimal control to a subinterval is optimal to the corresponding restricted optimization problem.*

An interpretation of this is shown in the schematic figure 2.1.

Using the principle of optimality, a derivation of the dynamic programming follows below. The steps follow [9] closely.

Assuming that each piecewise continuous control $u(\cdot)$ uniquely defines the evolution of the state, a *cost-to-go function*, $J(t, x, u(\cdot))$, is defined by

$$J(t_0, x_0, u(\cdot)) = \phi(x(t_f)) + \int_{t_i}^{t_f} f_0(t, x(t), u(t))dt.$$

The *optimal cost-to-go function* (or *value function*) is denoted

$$J^*(t_i, x_0) = \min_{u(\cdot)} J(t_i, x_0, u(\cdot)),$$

where the minimization is performed with respect to all admissible controls. The principle of optimality implies that

$$\begin{aligned} J^*(t_0, x_0) &= \int_{t_0}^{t'} f_0(s, x^*(s), u^*(s))ds + J^*(t', x^*(t')) \\ &= \min_{u(\cdot)} \left\{ \int_{t_0}^{t'} f_0(s, x(s), u(s))ds + J^*(t', x(t')) \right\}, \end{aligned}$$

which, with $t_0 = t$ and $t' = t + \Delta t$ becomes

$$J^*(t, x(t)) = \min_{u(\cdot)} \left\{ \int_{t}^{t+\Delta t} f_0(s, x(s), u(s))ds + J^*(t + \Delta t, x(t + \Delta t)) \right\}.$$

This is called the *dynamic programming equation*, and it recursively gives the solution to the optimal control problem. In its discrete form it offers a convenient way to find the optimal control and the corresponding optimal value of the problem.

The discrete optimal control problem can be formulated as

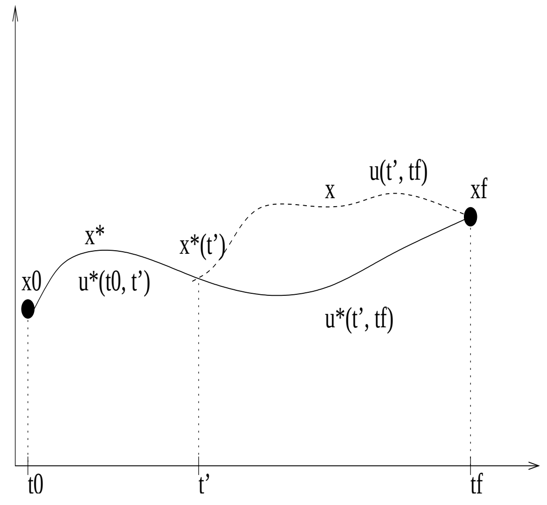


Figure 2.1: The path $x^*(\cdot)$, (solid line) obtained by using the control $u^*(\cdot)$ is optimal on the interval $(t0, tf)$. The principle of optimality then says that the path obtained by $u^*(\cdot)$ is optimal for the problem on the interval (t', tf) starting in the point $\{t', x^*(t')\}$. The dashed trajectory shows the path corresponding to another control on the interval (t', tf), which yields a higher cost.

$$\begin{array}{ll} \min & \phi(x_N) + \sum_{k=0}^{N-1} f_0(k, x_k, u_k) \\ \text{subj to} & \begin{cases} x_{k+1} = f(k, x_k, u_k), \quad u_k \in \mathcal{U}, \\ x_0 \in S_i, \\ x_N \in S_f. \end{cases} \end{array}$$

Since the optimal cost to go function is $J^*(N, x) = \phi(x_N)$ in the final state, the dynamic programming equation yields the recursion

$$J(N, x) = \phi(x),$$

$$J(n, x) = \min_{u \in U(n,x)} \{f_0(n, x, u) + J(n + 1, f(n, x, u))\}, n = N - 1, N - 2, \ldots, 0$$

The optimal value of the problem is given by $J(0, x_0)$ and the optimal control in each stage is

$$u_n^* = \mu(n, x) = \text{argmin}_{u \in U(n,x)}\{f_0(n, x, u) + J(n + 1, f(n, x, u))\}.$$

To find the optimal solution of a continuous problem is not quite as simple, since it involves solving a partial differential equation. This is however not treated further here, since dynamic programming is used only in its discrete form in this thesis work. For a survey of methods for systematically finding the optimal solution of a continuous problem, see [9] or [5].

2.2 Some Special Graph Problems

Some optimization problems have very special properties when defined on a network, which can simplify the solutions remarkably. In this section two network specific linear programming problems and their solutions are presented, but first a survey of some network notation is given.

2.2.1 Graph Notation A *graph*, $G = (E, V)$ is a set of nodes or vertices, $v_i \in V$, pairwise connected by edges, $e_{i,j} = (v_i, v_j) \in E$. In a *directed graph*, the edges are one-way, i.e. $e_{i,j} \neq e_{j,i}$ and sometimes called *arcs*. A *weighted* graph have costs, $c_{i,j}$, associated with the edges. These costs are often, but not always, constrained to be real and positive, e.g. when they represent the length of the edges. The *indegree* and *outdegree* of a node is the number of edges starting and ending in the node, respectively.

A *network* is a directed graph that has a set of *sources* (nodes with indegree 0), S, and a set of *sinks*, T (with outdegree 0), together with a *capacity*, $b_{i,j}$, for each edge. The *flow* through the network, f, is an array with one component, $f_{i,j}$, for each edge such that

$$\begin{array}{ll} 0 \leq f_{i,j} \leq b_{i,j} & \forall (i,j) : e_{i,j} \in E \\ \sum_{j:e_{i,j} \in E} f_{i,j} = \sum_{j:e_{j,i} \in E} f_{j,i} & \forall v_i \in V - S - T. \end{array}$$

An example of a network is shown in figure 2.2.

2.2.2 The Shortest Path Problem Given a directed graph (E, V) with nonnegative costs $c_{i,j}$ on the edges, the shortest path problem is the problem of finding the path $P(s, t)$ from node s to node t, that has the lowest total cost $\sum_{e_{i,j} \in P(s,t)} c_{i,j}$. This task can be solved in $\mathcal{O}(n^2)$ time by using Dijkstra's algorithm, [17]. This method produces an array of labels, one for each vertice v_i in V, that holds the length of the shortest path from s to v_i.

The algorithm starts by defining a subset W of V, that contains the node s alone. Next $\rho(v)$ is defined as the array containing the length of the shortest path from s to v_i, using only intermediate nodes contained in W. If there is no such path, $\rho(v_i)$ is set to ∞. The method propagates by, recursively, including the vertice with the lowest $\rho(v_i)$, not already in W, into W, and then recalculating the array ρ. The algorithm is finished when all vertices for which $\rho(v_i) \neq \infty$ are included in W. The array ρ then holds the shortest path from s to every node in V. The algorithm is illustrated in figure 2.3.

2.2.3 The Min-Cost Flow Problem Let $G = (V, E)$ be a directed graph with nonnegative costs $c_{i,j}$ on the arcs, that has m sources with a supply $a_i,\ i = 1 \ldots m$ and n sinks with a demand of $b_j,\ j = 1 \ldots n$. The min-cost flow problem is then

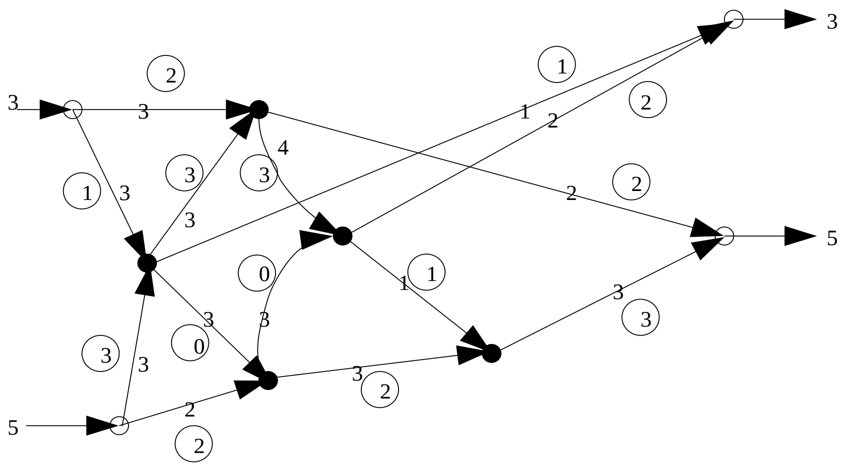


Figure 2.2: A network with two sources and two sinks. The capacities $b_{i,j}$ are shown as numbers on the edges. The flow $f_{i,j}$ through the edges is shown in circles. The flow coming in to a node must be equal to the flow leaving the node, except for at the sources and sinks, shown as small white circles.

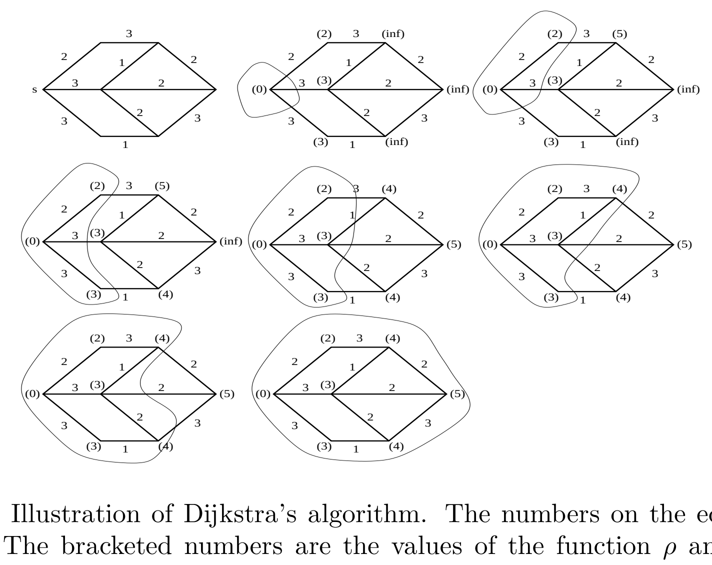


Figure 2.3: Illustration of Dijkstra's algorithm. The numbers on the edges are the costs $c_{i,j}$. The bracketed numbers are the values of the function ρ and holds the length of the shortest path from node s, using only previous nodes in W.

defined as finding the flow f_0, that satisfies the supply and demand of sources and sinks and that has the minimum cost, c_0.

The special case of the min-cost flow problem where the cost $d_{i,j}$ of sending a flow unit from source i to j is known, is called the Hitchcock problem. Stated as an LP-problem it looks like

$$\begin{array}{rll} \min & \sum_{i,j} d_{i,j} f_{i,j} & \\ \text{subj. to} & \sum_{j=1}^{n} f_{i,j} = a_i, & i = 1..m \\ & \sum_{i=1}^{m} f_{i,j} = b_i, & i = 1..n \\ & f_{i,j} \geq 0 & \end{array}$$

and can be solved with for instance the network simplex method or the alpha-beta algorithm, which is a so-called primal-dual method. It starts with a dual feasible solution and constructs a restricted primal problem. This has the form of a shortest path problem, and is easily solved. The solution to the restricted primal is used to improve the dual variables, until the optimal solution is obtained. A description of the algorithm can be found in [17].

Both the network simplex and the alpha-beta algorithm have exponential worst-case performance [16]. However, if $m = n$ and the supply and demand of the sources and sinks are all 1, the Hitchcock problem simplifies to the so-called *assignment problem.* In this case it is possible to show that the alpha-beta algorithm solves the problem in $\mathcal{O}(n^3)$ time, [17].

2.3 The Zero-sum Game

In control theory, the interest is often not only on *one* system, controlled by *one* agent, but on problems where multiple agents control a system of systems in various ways. A common approach to these problems is to formulate a performance index of the solution, which is optimized with respect to the controls of the different agents, as the cost function in section 2.1 above.

When the agents in question have conflicting or even diametrically opposite objectives for their control, the problem cannot be solved as an ordinary minimization problem. If there are only two agents acting for diametrically opposite objectives, one talks about a *two person zero sum game* [3]. The two agents are then called players and their controls are here denoted u and w, respectively. Since the objectives of the players are strictly opposite, i.e. one player wins whatever the other player looses, the performance index can be thought of as a function $\mathcal{F}(u,w)$, representing the amount that is lost and won by the two players.

A static two person zero sum game is constituted of this $\mathcal{F}(u,w)$ defined on the product vector space $\mathcal{U} \times \mathcal{W}$, minimized by $u \in U \subset \mathcal{U}$ and maximized by $w \in W \subset \mathcal{W}$.

The worst case scenario for the minimizing player is the outcome

$$\mathcal{F}^*(u) = \max_{w \in W} \mathcal{F}(u,w),$$

and the worst case for the maximizing player is

$$\mathcal{F}_*(w) = \min_{u \in U} \mathcal{F}(u,w).$$

The minimizing player now faces the so called 'min-max-problem'

$$\min_{u \in U} \mathcal{F}^*(u) = \min_{u \in U} \max_{w \in W} \mathcal{F}(u,w),$$

while the maximizing player faces the 'max-min-problem'

$$\max_{w \in W} \mathcal{F}_*(w) = \max_{w \in W} \min_{u \in U} \mathcal{F}(u,w).$$

These two problems are dual to each other and for them the weak duality statement

$$\mathcal{F}_*(w) = \min_{u\in U} \mathcal{F}(u,w) \le \mathcal{F}(u,w) \le \max_{w\in W} \mathcal{F}(u,w) = \mathcal{F}^*(u) \tag{2.1}$$

holds.

The optimal solution of the max-min-problem is thus bounded above by the optimal solution of the min-max-problem. If the inequality 2.1 above holds with equality, the game is said to have a *saddle point* (u_*, w_*) and $\mathcal{F}(u_*, w_*)$ is then the *value* of the game.

When looking at a dynamic system, with a state that propagates in time, the static zero-sum game can be extended to a dynamic zero sum game. Consider a discrete time dynamic system, where the players make the control decisions u_k and w_k in each time step $k = 1 \ldots K$, to minimize and maximize, respectively, the cost function

$$L(u,w) = \sum_{k=1}^{K} g_k(x_{k+1}, x_k, u_k, w_k).$$

The state x_k propagates according to some state equation

$$x_{k+1} = f_k(x_k, u_k, w_k).$$

As an analogue to the dynamic programming recursion (see section 2.1) that gives a solution to optimal control problems with only one agent, there are a set of recursive equations giving a sufficient condition for the existence of a saddle point of a two person zero sum game [3]:

$$\begin{aligned} V_k(x) &= \min_{u_k\in U} \max_{w_k\in W} [g_k(f_k(x,u,w),x,u,w) + V_{k+1}(f_k(x,u,v))] = \\ &= \max_{w_k\in W} \min_{u_k\in U} [g_k(f_k(x,u,w),x,u,w) + V_{k+1}(f_k(x,u,v))] \\ V_{K+1}(x) &= 0. \end{aligned}$$

For the equality of the first two rows to hold, the static game encountered in each time step must have a saddle point. If this is not the case, the recursion splits into two:

$$\begin{aligned} \overline{V}_k(x) &= \inf_{u_k\in U} \sup_{w_k\in W} [g_k(f_k(x,u,w),x,u,w) + \overline{V}_{k+1}(f_k(x,u,v))] \\ \overline{V}_{K+1}(x) &= 0 \end{aligned}$$

and

$$\begin{aligned} \underline{V}_k(x) &= \sup_{w_k\in W} \inf_{u_k\in U} [g_k(f_k(x,u,w),x,u,w) + \underline{V}_{k+1}(f_k(x,u,v))] \\ \underline{V}_{K+1}(x) &= 0, \end{aligned}$$

with $\overline{V}_1$ and $\underline{V}_1$ being the upper and lower values of the dynamic game. If there exists a sequence $\bar{\mu} = \{\bar{\mu}_k\}_{k=1}^{K}$, such that $u = \bar{\mu}_k(x)$ gives the infimum in the $\overline{V}_k(x)$-equation, then $\bar{\mu}$ is called a *minimax* policy for the minimizing player. Symmetrically, a *maximin* policy for the maximizing player is given by the $\underline{V}_k(x)$-equation [3]. If there is a saddle point in the dynamic game, then the minimax- and the maximin policies are saddle point policies.

2.4 The Unicycle and Dubins' Car

In this section the model chosen for the UAVs is described. This model is used in many applications and can be adjusted to capture the kinematics of both ground vehicles and aerial robots.

2.4.1 The Unicycle Model A simple unicycle model represents a robot that can turn around freely but only move in the direction of its orientation. Letting z_1, z_2 denote the position of the robot with orientation θ, its motion is described by

$$\begin{aligned}\dot{z}_1 &= v\cos\theta,\\ \dot{z}_2 &= v\sin\theta,\\ \dot{\theta} &= \omega,\end{aligned}$$

where the inputs v and ω denotes lateral and angular velocity.

Introducing constraints on the controls, such as $v \in [v_{min}, v_{max}]$, $\omega \in [-v_{turn}, v_{turn}]$, where $v_{min},\ v_{max},\ v_{turn} > 0$, the unicycle model can be used as a rough representation of more complicated vehicles, e.g. the unmanned aircraft studied in this thesis.

2.4.2 Dubins' Car The problem of Dubins' car is to find the shortest path from the initial position $\underline{x}_i = [x_{1,i}, x_{2,i}, \theta_i]$ to the final position $\underline{x}_f = [x_{1,f}, x_{2,f}, \theta_f]$ for a unicycle robot with the kinematics

$$\begin{aligned}\dot{x}_1 &= v\cos\theta\\ \dot{x}_2 &= v\sin\theta\\ \dot{\theta} &= \omega,\end{aligned}$$

that is travelling at constant speed v. Letting T be the time when the vehicle has reached $\underline{x}_f$ the Dubins' car problem is equivalent to the following optimal control problem.

$$\begin{array}{ll}\min & T = \int_0^T \mathrm{d}t \\ \text{subj to} & \left\{\begin{array}{l}\dot{x}_1(t) = v\cos\theta(t)\\ \dot{x}_2(t) = v\sin\theta(t)\\ \dot{\theta} = \omega(t)\\ [x_1(0), x_2(0), \theta(0)] = \underline{x}_i\\ [x_1(T), x_2(T), \theta(T)] = \underline{x}_f\\ |\omega(t)| \le v/R\end{array}\right.\end{array}$$

The control is restricted since a vehicle travelling at constant speed without slipping will experience a minimum turning radius, R, so that the angular velocity is bounded to $|\omega(t)| \le v/R$.

The optimal control problem can be solved using the Pontryagin minimum principle. This method is described in most introductory literature on optimal control, and a good exposition can be found in [9]. It gives a necessary condition for optimality by locally investigating the optimal control and state trajectories. Defining the *Hamiltonian function*, $H(x, u, \tilde{\lambda})$ as

$$H(x, u, \tilde{\lambda}) = \tilde{\lambda}^T \begin{bmatrix} f_0(x, u) \\ f(x, u) \end{bmatrix},$$

the principle is described by the following theorem.

Pontryagin Minimum Principle If $(x^*(t), u^*(t), t_f^*)$ is an optimal solution to the optimal control problem with n state variables, then there exists a nonzero extended adjoint function $\tilde{\lambda}(\cdot) = [\lambda_0\ \lambda_1 \ldots \lambda_n]$, such that

(i)

$$\dot{\tilde{\lambda}}(t) = -\frac{dH(x^*(t), u^*(t), \tilde{\lambda}(t))}{d\tilde{x}}$$

(ii)

$$u^*(t) = \text{argmin}_{u \in U} H(x^*(t), u, \tilde{\lambda}(t)),\ \forall t \in [0, t_f^*]$$

(iii)

$$H(x^*(t), u^*(t), \tilde{\lambda}(t)) = 0,\ \forall t \in [0, t_f^*]$$

(iv)

$$\lambda_0(t) = \text{constant} \geq 0.$$

In the Dubins' car problem the Hamiltonian is

$$H(\underline{x}, \omega, \tilde{\lambda}) = \lambda_0 + \lambda_1 v \cos\theta + \lambda_2 v \sin\theta + \lambda_3 \omega.$$

According to *(ii)*, pointwise minimiziation of this with respect to ω gives the optimal control

$$\omega^* = \text{argmin}_{|\omega| \leq v/R} H(\underline{x}, \omega, \lambda) = \text{argmin}_{|\omega| \leq v/R} \lambda_3 \omega,$$

i.e. the minimizing control is v/R (or *turn left*) for $\lambda_3 < 0$ and $-v/R$ (or *turn right*) for $\lambda_3 > 0$. To find the control for the case when $\lambda_3 = 0$ during a nonzero time interval, the equations *(i)* and *(iii)* are used. First *(i)* gives the *adjoint equations*

$$\begin{aligned} \dot{\lambda}_1(t) &= 0 \\ \dot{\lambda}_2(t) &= 0 \\ \dot{\lambda}_3(t) &= \lambda_1(t) v \sin\theta(t) - \lambda_2(t) v \cos\theta(t) \end{aligned}$$

that shows that λ_1 and λ_2 are constants. Then equation *(iii)* becomes

$$\lambda_1 v \cos\theta(t) + \lambda_2 v \sin\theta(t) = -\lambda_0$$

on time intervals where $\lambda_3 = 0$. This implies that $\theta(t)$ is constant on such interval, and the control $\omega(t)$ thus have to be 0.

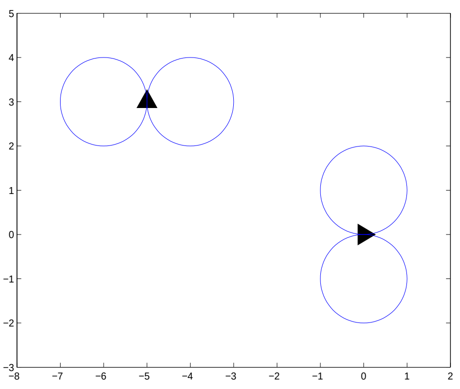
The optimal solution to the Dubins' car problem is

$$\omega^*(t) = \begin{cases} v/R, & \lambda_3(t) < 0, \\ 0, & \lambda_3(t) = 0, \\ -v/R, & \lambda_3(t) > 0 \end{cases}$$

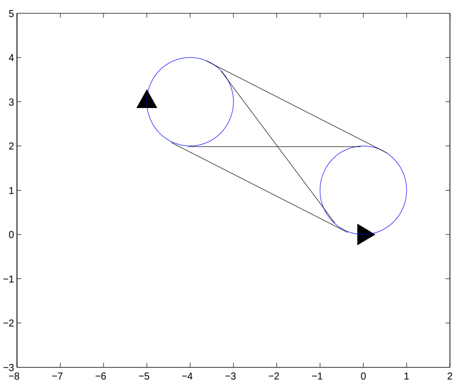
which corresponds to the trajectory turning left with turning radius R, following a straight line or turning right with radius R, respectively. The optimal path is thus a combination of left turns, right turns and straight lines.

This is as far as the Pontryagin minimum principle will bring the solution. To find the sequence and lengths of the segments some other method must be used. In 1991 Sussman and Tang presented a geometrical means for finding the optimal solution to the problem. This is presented in [2], together with an algorithm to synthesize the optimal path. It is briefly described here.

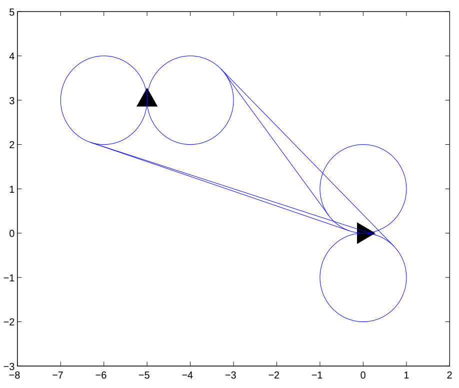
The main result is that the optimal path is on the form $B_a S_b B_c$ or $B_d B_e B_f$, where B denotes a curved segment, either a right- or a left turn, and S denotes a straight line. The subscripts indicate the length of the segment and obey certain absolute and relative constraints. The lengths of the segments are of course non-negative, but one



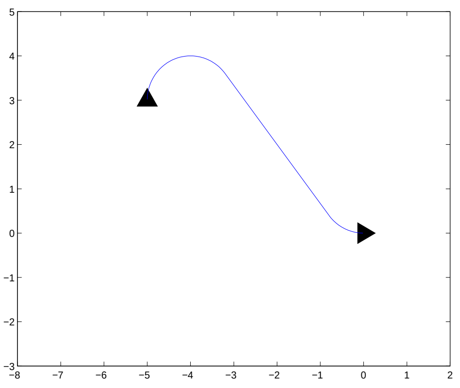
(a) Two circles of minimum turning radius are formed around the initial and final positions



(b) Each of the initial circles can be connected to each of the final circles by four tangents.



(c) The circles of the UAV's minimum turning radius are connected by four feasible paths among which one is the shortest path.



(d) A simple comparison between the four connecting paths yields the shortest path.

Figure 2.4: The four steps in the process of finding the shortest path on the form BSB.

or two of them might equal 0, the path being composed of less than three segments. The three steps in an algorithm for finding the shortest path on BSB-form are given below.

Step 1. Two circles determined by the minimum turning radius are constructed around the initial and final positions, as is shown in figure 2.4(a).

Step 2. From any of the initial circles it is possible to construct four tangent connections to any of the final circles (see figure 2.4(b)). Combined with segments of the start- and end circles, only one of these four gives a path smooth enough to be feasible for the robot. A set of three equations is used to find the correct tangent line for each pair of circles.

Step 3. There are now four paths on the form BSB among which the shortest path from $\underline{x}_i$ to $\underline{x}_f$ is to be found (figure 2.4(c)). The lengths of the four paths is calculated, and a simple comparison between them gives the optimal path, seen in figure 2.4(d).

3. Timing Tactics Against a Single SAM Opponent

The subproblem of timing considers the behaviour of the UAV as it approaches an enemy SAM site on its path towards the target.

As the UAV comes closer to the fixed SAM site the two can act and interact in several ways. At every instant the UAV can fire a HARM at the SAM site, which on its part can choose to launch a SAM or to shut down its radar. The UAV may react on a launched SAM by turning back, trying to take another path around the obstacle, or by climbing to a higher altitude. Other actions may be jamming or adapting the speed to the given situation. Since there is no pilot to make these decisions, the UAV must be provided with an action strategy that gives a good result regardless of what the SAM site will do.

3.1 Problem Formulation

A scenario where a UAV carrying m available HARMs approaches a stationary SAM site along a straight line trajectory is considered. Both SAM site and UAV are allowed to take a restricted set of actions. The SAM site may turn on or off its radar, or fire a SAM at the approaching UAV. The UAV on the other hand has the choice to fire one of its HARMs or not. The threat from the SAM site to the UAV and vice versa is expressed with probabilities. When the SAM site has launched a missile, the probability that it will be lethal to the UAV is given by $p_s(d)$, depending on the distance d between the SAM site and the UAV at the time of intercept. If, for example, the UAV is beyond the range of the missile at the time of intercept, the missile will miss. Analogously, the probability that a fired HARM will hit the SAM site, $p_h(d)$, is a function of the distance between the HARM and the SAM site at the moment when the threatened site turns off its radar. If, e.g., the HARM is very close to the SAM site when the radar is turned off, there is still a high probability that the radar will be destroyed.

The probabilities $p_s(d)$ and $p_h(d)$ are given by

$$\begin{aligned} p_s(d) &= (1 - S(d, r, s_1)) \cdot S(d, 0.1r, s_2) \cdot S(arcsin(h/d), g, s_3) \\ p_h(d) &= 0.8 \cdot 10^{\frac{-d}{10}}. \end{aligned}$$

where h is the altitude of the UAV, r is the SAM site's radar range, the constants s_1, s_2 and s_3 are parameters to be tuned in to realistic values in a simulation, and the function $S(x, x_0, s)$ used in $p_s(d)$ is a soft step function given by

$$S(x, x_0, s) = \frac{1}{2} \cdot (1 + \frac{x - x_0}{\sqrt{(s^2 + (x - x_0)^2)}}).$$

A SAM site that turns off its radar is out of reach of the UAV's radar warning receiver (RWR). This is the reason why the HARM hit probability depends on the position of the HARM at the shutdown of the SAM site. It indicates the distance the HARM has to go without guidance, trusting the bearing information received before shutdown. After the SAM site has turned off its radar there is a startup delay before it can be up and running again. Furthermore, all fired SAMs are lost at a shutdown, since they are guided by information collected by the SAM site's tracking radar. A shutdown thus yields a period of passivity for the SAM site, not threatening the UAV in any way, while HARMs fired before the shutdown might still be in the air, posing a threat to the SAM site, however, less efficiently than in the case of an up-and-running SAM site.

The stated problem is now to find out at what instants during the approach to the SAM site it is most favorable for the UAV to fire its HARMs. The SAM site is supposed to act logically according to the information available, to maximize its own benefit.

3.1.1 Modelling the System The problem described above is quite complex and in order to make it computationally treatable some further simplifying assumptions are made. First, turning back and fleeing is not considered as an option for the UAV. This is a serious limitation, but treating this case is beyond the scope of this thesis.

Secondly, time and space are discretized. By assuming that the UAV follows the specified path and stays on constant altitude, space can be considered one dimensional. In a model where the UAV is bound to a constant speed, time and location of the UAV can be handled together and the dimensionality of the problem thus be further reduced.

State model Let the system consisting of UAV and SAM site be represented by the state vector

$$\mathbf{x} = \begin{bmatrix} s \\ sl \\ hl_1 \\ hl_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} \text{state of SAM site \{on/off\}} \\ \text{time to detonation of SAM} \\ \text{distance from HARM no 1 to SAMsite} \\ \text{distance from HARM no 2 to SAMsite} \\ \vdots \end{bmatrix}$$

At each time/location decisions have to be made. The decision made by the UAV is represented by the control $\mathbf{u}$ while $\mathbf{v}$ are the actions taken by the SAM site.

$$\mathbf{u}(t) = \begin{bmatrix} u_1 \end{bmatrix} = \begin{bmatrix} \text{fire HARM \{yes/no\}} \end{bmatrix}$$

$$\mathbf{v}(t) = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \text{turn on/off radar \{on/off\}} \\ \text{fire SAM \{yes/no\}} \end{bmatrix}$$

The state vector propagates in time, which is the same as location of the UAV, according to the state equations $\mathbf{x}(t+1) = f(t, \mathbf{x}(t), u(t), \mathbf{v}(t))$. These are quite complex, and are for the sake of clarity presented in appendix A.

3.2 Proposed Solution

The situation with two actors controlling the state of the system independently with separate decisions, both striving to reach a desired situation (The UAV to pass the SAM site unharmed and the SAM site to remain intact and to prevent the UAV to pass), is natural to think of as a gaming situation. *Zero sum games* are described in section 2.3.

3.2.1 Modelling the Game The object function used in this timing problem should reflect the UAV's wish to pass the SAM site unharmed, as well as the SAM site's wish to remain intact and to prevent the UAV to pass. Since these goals are not strictly opposite this is not intuitively a zero-sum game. To simplify the computations, achieving a zero-sum game, it is however assumed that the UAV would gain more from a hit of the SAM site than from a shut down. This can be made plausible by considering the return of the UAV to the home base. A killed SAM site would be safe to pass at the return, while one that was just shut down will have had time to recover, and must be suppressed again on the way back. The object function is chosen as

$$\mathcal{F}(u, v) = \kappa P(\text{SAM hit}) - \lambda P(\text{HARM hit}) + \mu S.$$

where $P(\text{SAM hit})$ and $P(\text{HARM hit})$ are the probabilities that the SAM will hit the UAV and that the HARM will hit the SAM site, respectively. These are functions depending on the decisions, (u and v), continously made by the UAV and the SAM site and on the state of the system. If the SAM or the HARM are launched, the probabilities are given by $p_s(d)$ and $p_h(d)$ defined in the problem formulation above. The third term depends only on the final state of the system, and reflects the wish of the UAV to force the SAM site to shut down and the SAM site's strive to avoid it. This S denotes the state of the SAM site in the moment when the UAV passes over it. The constants κ, λ and μ are tuned to get a realistic, non trivial solution.

With this object function the UAV will be the minimizing player while the SAM site will be the maximizing. The strategy of the UAV would be the optimal solution, u^*, of the min-max-problem $\min_u \max_v \mathcal{F}(u, v)$.

3.2.2 Solving the Min-max-problem Dynamic programming is used to solve the min-max-problem. This is a standard technique when it comes to treating dynamic zero-sum games. Using the formulation in section 2.1, the discrete optimal control problem becomes

$$\min_{u(\cdot)} \quad \{\max_{v(\cdot)} \mu S + \textstyle\sum_{k=0}^{N-1} \kappa p_s(d(\mathbf{x})) + \lambda p_h(d(\mathbf{x}))\}$$

$$\text{subj. to} \quad \begin{cases} x_{k+1} = f(k, \mathbf{x}(k), \mathbf{u}(k), \mathbf{v}(k)), \\ x_0 \in S_i, \\ x_N \in S_f, \\ \mathbf{u}(k) \in \mathcal{U} \\ \mathbf{v}(k) \in \mathcal{V}. \end{cases}$$

The initial set S_i is the single state x^0, in which no missiles are fired and the SAM site radar is on. The final set contains all states with combinations of missiles that have reached their targets or are not fired yet, and where the SAM site is on or off. The control constraints are $\mathcal{U} = \{0, 1\}$ and $\mathcal{V} = \{[0, 0], [0, 1], [1, 0], [1, 1], \}$. The solution to the problem will be a sequence of control sets, both for the UAV and for the SAM site, one for each time step. The discrete optimal control problem is illustrated in figure 3.1.

Solving both the primal min-max-problem and the dual max-min-problem shows that in this case there is no saddle point (u^*, v^*) rendering a common optimal value of the two problems. The min-max-problem is a model of the situation when the SAM site can observe the actions made by the UAV. The SAM site is probably manned and will always react on the actions of the UAV to maximize the outcome of the target function. The UAV's strategy must therefore be chosen to contain the actions that give the smallest outcome when the SAM site has maximized it. The min-max-problem is hence the more realistic of the two.

The success rate of the mission is in this case chosen as the probability that the UAV will pass the SAM site unhurt. This is computed as the complement of the probability that the SAM hits the UAV, given that the SAM site is unharmed at the moment of detonation. The success rate in timestep t, $p_{SR,t}$, is thus given by

$$\begin{aligned} p_{SR,t} &= 1 - P(\text{UAV is hit in step} t | \text{UAV has not been hit}) = \\ &= 1 - p_{st} \prod_{t_i \le t} p_{ht_i}, \end{aligned}$$

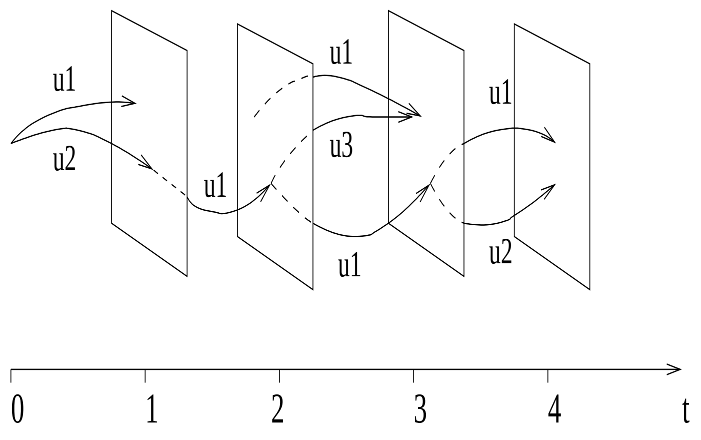


Figure 3.1: Illustration of the dynamic programming setup. The planes at each timestep represents the set of possible states $\mathcal{X}(t)$. Between each set of states a number of transfers are possible, here a subset of them are shown as arrows. The control yielding each transfer is denoted by ui, representing the different controls $(\mathbf{u}, \mathbf{v})$. The optimal control sequence is obtained by backwards recursion, as is described in section 2.1.

and the total succes rate is obtained by

$$p_{SR} = \prod_{\forall t} p_{SR,t}.$$

Complexity The state of the SAM site, $\mathbf{x}(1) = s$, is binary and gives rise to 2 possible states. The distance from the SAM to the UAV, as well as the distances from the HARMs to the SAM site can take all values between 0 and the time t. This gives t^{h+1} states, where h is the number of HARMs used at the SAM site. At every time step t the number of possible states of the system is $\mathcal{O}(t^{h+1})$. The number of additions performed at every step of the dynamic programming recursion is $\mathcal{O}(t^{h+1})$ and the number of comparisons are $2t^{h+1} \cdot 2^3$, so the complexity of the solution with dynamic programming in T steps is $\mathcal{O}(T^{h+2})$.

3.3 Simulations

Simulations have been done for a variety of scenario set-ups, with the time window, i.e. the distance from the UAVs initial position to the SAM site, varying from 15 km (which is the range of an SA-8 SAM site) up to 60 km. All simulations were performed with only one SAM (except for some with a very short time window), since the simulations environment (matlab) yielded highly time consuming computations. Performing the simulations in e.g. C++, and using the appropriate data structures would render slightly more complex situations computationally feasible. The result from a simulation using one HARM against an SA-8 unit, starting from 30 km is shown in figure 3.2. The success rate of this solution is 0.77. Even though the simulation shows a hit of the UAV at time step five, the probability for this hit is only 0.2, and it is favorable for the UAV to take this risk of being hit, and fire its HARM at a closer distance.

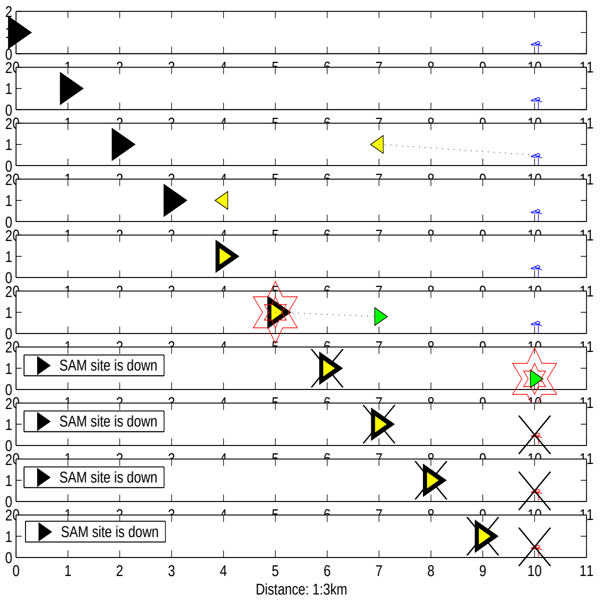


Figure 3.2: The proposed solution to the timing problem for a UAV carrying one available HARM and approaching an SA-6 air defence unit. The subplots illustrates the behavoiur of UAV and SAM site at ten successive timesteps at 0.01 s interval. The stars around SAM site and UAV in steps 6 and 7 shows detonations of SAM and HARM, resulting in a possible hit of the UAV and SAM site, respectively. The success rate using this strategy is 0.77.

3.4 Conclusions

The proposed solution gives a hint of how to distribute the HARMs optimally. The firing schemes sometimes seems to prescribe high risks for the UAV, as the delay of shooting in figure 3.2. The risks are however weighed against the losses of shooting earlier. Integrating this method in a pathplanning phase, using the success rates obtained here as input when choosing paths would give a good estimate of the threat along the paths. It would also offer the possibility to turn around and flee, which was ignored in this approach.

The optimization gives the optimal controls for both UAV and SAM site, and the success rate of the solution builds on the assumption that the SAM site follows this rational control scheme. If, however, the SAM site would use some other strategy, the result would be better, since the choices of the UAV were made to optimize the worst case situation.

4. Path Planning and Target Assignment

One critical part of a SEAD mission is the mission planning. This task includes finding the best path through the area covered by the air defence. Since the threat posed to a UAV as it passes a SAM site is highly dependent on whether the UAV opens fire at the threat or not, the path planning is tightly coupled with the task of target assignment, i.e. the search for a strategy to assign certain SAM sites as targets, while others are being ignored.

This chapter will initially handle the two subjects separately, and then propose an integrated solution to the two problems.

4.1 Problem Formulation

To study techniques of finding the safest path, a specific scenario is considered. This scenario contains a number of communicating SAM sites of two different radar ranges placed around the target of the SEAD mission. Also included in the scenario is a group of five UAVs used to execute the mission. One of the five UAVs does not carry any air-to-ground missiles, but is protected by the rest of the SEAD force on the path towards the target. The SAM sites are placed so that no safe corridor exists from the initial position of the UAVs to the target area. The area considered in the scenario is limited to a square around the target and the UAVs' initial positions. The size of this square is determined by the amount of available fuel.

The SAM sites are characterized by the threat that they pose to the UAVs, and are in this scenario represented by the probability densities $p_{l,i}(h,d)$ and $p_{s,i}(h,d)$. These are the probability that SAM site i would launch a SAM at a UAV at altitude h and distance d from the site, and the probability that a launched SAM would hit its target, located at (h,d) at the time of intercept, respectively. The probabilities are given by

$$\begin{aligned} p_{l,i}(h,d) &= (1 - S(d,R,s_1)) \cdot S(d, 0.1 \cdot R, s_2) \cdot S(arcsin(h/d), \gamma, s_3) \\ p_{s,i}(h,d) &= (1 - S(d,r,s_4)) \cdot S(d, 0.1 \cdot r, s_5) \cdot S(arcsin(h/d), \gamma, s_6) \end{aligned}$$

where R is the radar range of the SAM site and the parameters s_1 to s_6 are parameters to be tuned in to realistic values in a simulation. The function S is a soft step function given by

$$S(x, x_0, s) = \frac{1}{2} \cdot (1 + \frac{x - x_0}{\sqrt{(s^2 + (x - x_0)^2)}}).$$

As opposed to the case in chapter 3, the SAM sites are in this scenario substituted by the probabilities above, and no concern is taken to capture their strive to maximize their own benefit, i.e. the decisions of the SAM sites are exchanged for the probabilities above.

The threat posed to a UAV from a SAM site is discrete in time. At every time instant the SAM site might launch a SAM, and the probability for this event was introduced above.

On their way towards the target, the UAVs can reduce the threat by firing the available HARMs. This procedure may either kill the targeted SAM site, or force it to shut down its radar to avoid the attack. Since there are generally not enough HARMs to attack all the passed SAM sites, it is important to prioritize the sites, assigning HARMs only to those contributing most to the overall threat. It is also desirable not to assign all available HARMs, in order to be prepared for any a priori unknown SAM sites, so called pop up-threats. The prioritization of the SAM sites should weigh the threat posed on the UAVs when a HARM is fired against the threat when the UAVs try to pass without firing a HARM.

4.2 Proposed Solution

To find the solution to the path planning and assignment problem, it is convenient to first handle the two subjects separately, to introduce the necessary simplifications and restrictions. In section 4.2.1 a path from the initial position, through the dangerous area covered by air defence units to the target is sought for. In section 4.2.2 a the prioritization of the SAM sites is handled. Then, in section 4.2.3 , a combination of the two is proposed. In this work the five UAVs are supposed to fly in some appropriate formation, thus basically following the same path.

The proposed solution presents an algorithm that produces waypoints for the UAVs and a set of SAM sites that should be fired at. The algorithm can be summarized in the 5 following steps:

1. *Form a three dimensional grid composed of the edges shown in table 4.1 in section 4.2.3 below.*
2. *To each edge, assign the cost stated in equation (4.1).*
3. *Perform a shortest path-search in the grid, using i.e. Dijkstra's algorithm.*
4. *From the result of the search, extract waypoints to follow and the set of SAM sites to fire at.*
5. *Smoothen the path to get rid of unnecessary coarsities, as described in section 4.2.4.*

Below follows a description of the ideas and simplifications used, as discussed above.

4.2.1 Path Planning Since searching a three dimensional space for an optimal trajectory in this case is a non convex infinite dimensional task, the dimensionality of the problem has to be reduced. This may be done using a Voronoi diagram, [7] , constructed around the known SAM sites. This is a commonly used approach, which can be found in e.g. [4] and [14] , to produce a set of waypoints that can then be used to construct a smooth, feasible trajectory. The Voronoi diagram partitions the horizontal plane into n convex cells, one for each SAM site, such that the distance from a point p in the cell to the SAM site of this cell is shorter than the distance to any other SAM site. The points on an edge separating two cells are thus equidistant from the two closest SAM sites and, conversely, the distance to the closest SAM site is maximized on the edge. Finally, the target and the initial UAV locations are also contained in cells in the Voronoi diagram, with extra edges connecting them with the vertices of the cell, as is depicted in figure 4.1.

When the Voronoi diagram has been constructed, weights are assigned to each edge, reflecting how dangerous it would be to fly along that edge. The Voronoi diagram can now be considered as a graph, in which the shortest path from one point to another can be found by e.g. *Dijkstra's algorithm* (see section 2.2.2).

Using paths composed of Voronoi-edges eliminates trajectories close to the threats, but does not consider the fact that the SAM radars may have different ranges (and thus threats). In some cases it might be favorable to deviate from the edges of the

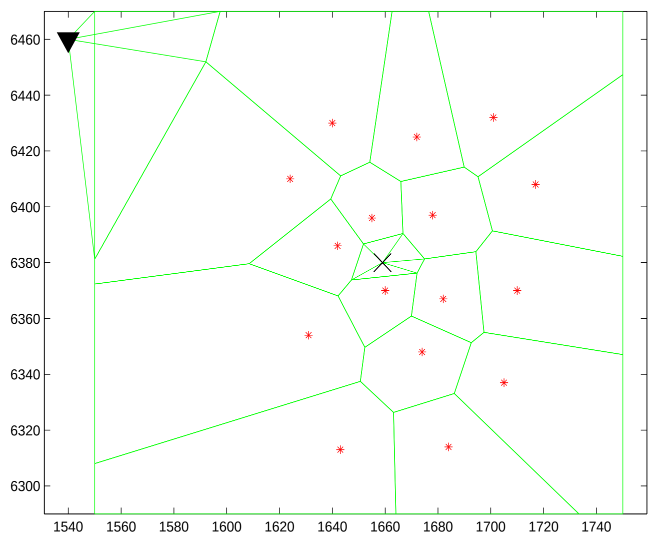


Figure 4.1: A Voronoi diagram constructed around a number of SAM sites and two additional sites (initial and final locations of the UAVs, marked as a triangle and a cross). The two latter are connected to the vertices of their Voronoi cells.

Voronoi diagram, approaching one threat but leaving the engagement area of another. For the purpose of taking advantage of these possible reductions in threat, additional edges are added that approximately follows the borders of the radar ranges. An example of such edges is shown in figure 4.2

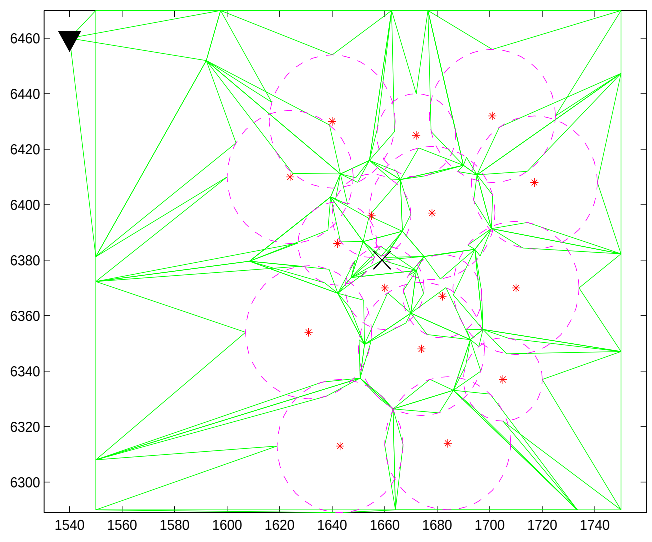


Figure 4.2: The Voronoi grid shown in figure 4.1 is here extended with extra edges approximately following the borders of the SAM sites. Where the SAM sites are situated close to each other this may result in edges of remarkably lower cost. In the outskirts of the area covered by the SAM sites, the extra edges doesn't even closely follow the radar ranges, and will neither be of much use, nor make any harm.

Assigning weights to the edges In order to be able to compare the paths, costs must be assigned to all possible path segments, i.e. edges. The cost associated with edge i is denoted $J_{tot}(i)$. This cost is a weighted combination of the length of the edge, L_i, and the threat exposure on the edge, $J_{threat}(i)$. The nature of the threat can vary depending on the definition of the mission. It may be the probability of being detected by the enemy, or the probability of crashing into the terrain, as a function of the topology. Here the threat is derived from the probability of being killed when flying along the edge, p_k. This probability depends on the probability that a SAM is launched, p_l, and the probability that this missile hits the UAV, p_s.

$$p_k(x) = p_l(x)p_s(x + \delta x).$$

The variable x is the position of the UAV and $x+\delta x$ is the position at time of intercept of the SAM. This position can be approximated from the planned path and speed of the UAV, and the *estimated time of arrival*,

$$T_{eta} = \frac{\|x - x_{SAMsite}\|}{v_{SAM}}.$$

$x_{SAMsite}$ denotes the position of the SAM site.

The launch- and hit probabilities $p_l(x)$ and $p_s(x)$ were introduced in the problem formulation and refer to the situation when the UAVs are in the radar range of one SAM site. Since the SAM sites are communicating parts of an integrated air defence system, IADS, there are however synergy effects increasing the threat when several SAM sites track a UAV at the same time. Both launch of a missile and hit of the target are more probable, due to increased information. The total probabilities are modeled as the union of the individual probabilities of the covering SAM sites.

$$p_{k,tot}(x) = \left(1 - \prod_{i \in S}(1 - p_{l,i}(x))\right) \cdot \left(1 - \prod_{i \in S}(1 - p_{s,i}(x + \delta x))\right),$$

where S is the set of SAM sites covering the UAV at position x and $p_{l,i}(x)$ and $p_{s,i}(x)$ are the launch and hit probabilities based on the threat from SAM site i. In the following, $p_k(x)$ is used to denote this total kill probability at position x.

The functions $p_l(x)$ and $p_s(x)$ have one value at every point x on the edge. There are infinitely many points and an approximation has to be done in order to make a calculation of the total threat on the edge possible. Since it takes some time to launch a SAM, making the decision of launching, giving the orders and pressing the buttons, the reduction of the number of instants for a possible launch to one per second may be reasonable. There are thus

$$g = \frac{l}{v_{UAV}}$$

possible launching instants while flying along an edge with length l.

Inspired by the approximation in [4], the sequence of possible launching instants are divided into six intervals of equal length, called $G1$ to $G6$. Letting x_j denote the position of the UAV at instant j, and x_{Gi} the centre point of interval Gi, the launch and hit probabilities are approximated to the values at the middle of each interval, $p_l(x_t; t \in Gi) = p_l(x_{Gi}))$ and $p_s(x_t; t \in Gi) = p_s(x_{Gi}))$.

Since the probability of being killed at some instant while flying along the edge is the probability of the union of the events *getting killed* at every instant, the total kill probability on edge i is

$$\begin{aligned} p_{ki} &= 1 - \prod_{t\in[1,g]} (1 - p_k(x_t)) = \\ &= 1 - [(1 - p_k(x_{G1}))(1 - p_k(x_{G2})) \cdots (1 - p_k(x_{G5}))(1 - p_k(x_{G6}))]^{g/6}. \end{aligned}$$

Since the kill probabilities of flying along separate edges don't sum up to the total kill probability of flying the sequence of them, p_{ki} is not a suitable measure of the threat on the edges. Given a set of edges A, the total kill probability of flying the sequence of all the edges in A is given by

$$p_{k,A} = 1 - \prod_{j\in A}(1 - p_{kj}).$$

The survival probability on the sequence of edges is

$$p_{\text{survive},A} = 1 - p_{k,A} = \prod_{j\in A}(1 - p_{kj}) = \prod_{j\in A} p_{\text{survive},j}.$$

But in order to use a shortest path algorithm such as the one described in 2.2.2, a sum to be minimized is needed.

Taking the logarithm of the survival probabilities and *changing sign* yields a sum ready to be minimized, using e.g. the shortest path algorithm.

$$-ln(p_{\text{survive},A}) = \sum_{j\in A} -ln(p_{\text{survive},j}).$$

The threat exposure on edge i is thus

$$J_{threat}(i) = ln(\frac{1}{1 - p_{ki}})$$

and the cost on the edge is

$$J(i) = \alpha\ L_{e_i} + \beta\ ln(\frac{1}{1 - p_{ki}}). \tag{4.1}$$

The weights α and β are set to achieve a proper balance between the often conflicting properties shortness and safety of the path. The total cost of flying a path P composed of the edges $e_i \in P$ is then

$$J_P = \alpha \sum_{e_i\in P} L_{e_i} + \beta \sum_{e_i\in P} ln((1 - p_{ki})^{-1}).$$

These costs offer the possibility to choose which path to fly. The next task is to investigate what SAM sites to engage with HARMs.

4.2.2 Target Assignment To prioritize the importance of firing at a SAM site, the event of passing a site without firing at it is given a weight. This weight is based on the total threat from the site at all edges along the path. The event of firing at the site when passing it is given the weight 0, indicating that it is safe to pass a SAM site that was fired upon. In a more accurate model the risk of missing the target should be taken into account. A cold-blooded enemy might refuse to turn off the radar, having a quite probable death before him, and thus, in case of a miss, being able to threaten the UAVs as effectively as if the HARM was not fired. To model the missile assignment as preferred, with a number of unassigned HARMs saved for potential pop up-threats, a weight is associated with the number of unused HARMs.

The problem as formulated above is visualized in figure 4.3. The optimal assignment plan is obtained with dynamic programming, which is described in section 2.1.

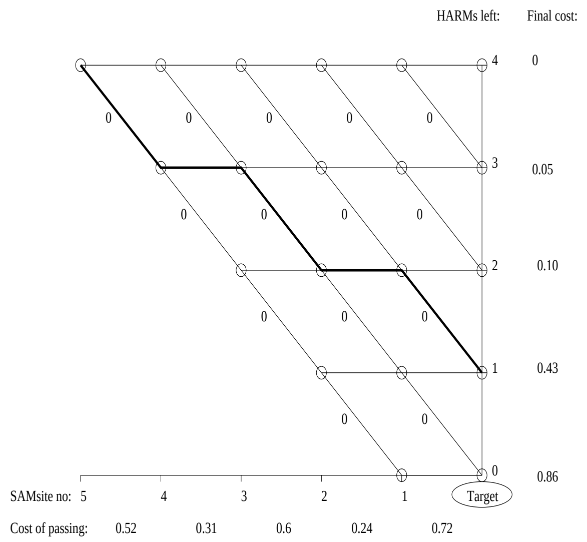

HARMs left:
Final cost:
4
0
3
0.05
2
0.10
1
0.43
0
0.86
SAMsite no: 5 4 3 2 1 Target
Cost of passing: 0.52 0.31 0.6 0.24 0.72
0

Figure 4.3: The dynamic programming recursion gives the solution to shoot at SAM site no 5, 3 and 1.

4.2.3 Combining Path Planning and Target Assignment Combining the path planning and the target assignment will give a more accurate solution to both problems. Before the details of such a combination are given, an example where the division will produce a suboptimal solution is described.

The least dangerous path computed in section 4.2.1 may be lined with small but numerous threats, in all not too dangerous, but too many to attack them all with the available HARMs. It might then be better to spend the few HARMs on board the UAVs on a couple of high-risk threats, to get a free way through the area. The optimal solution to the realistic problem will obviously not be found by giving the optimal path obtained in section 4.2.1 as input to the HARM assignment in section 4.2.2.

To reduce the loss of realism that occurs in the problem division, the target assignment and the path planning are combined at the cost of simplicity. The following scheme is proposed: The Voronoi grid constructed in section 4.2.1 is expanded to three dimensions. Identical graphs, calculated as in section 4.2.1, form a pile of planes, and between the planes additional connecting edges are added. The vertical level defines the number of available HARMs on the UAVs performing the mission, i.e. each vertex in the new graph is associated with a point $(r, n) \in R^2 \times N$, where r is the position and n is the number of HARMs. Firing a HARM at a SAM site is represented by descending one step in the three dimensional grid. Assuming that the shooting of a HARM neutralizes the SAM site, the connections between the 'floors' stretch over the Voronoi zone of the hit SAM site, as is illustrated in figure 4.4. In this way the model captures that firing not only reduces the threat, but also often shortens the path to go.

The descending edges that follow the border of a Voronoi zone represents the firing at any of two SAM sites (the sites of the zones that are separated by the edge). Using this edge, the UAV is supposed to have shot at the site contributing most to the threat.

Note that the edges stretching between the planes are directed edges, since it is impossible to go back to a higher level 'earning' a HARM.

There are now edges stemming from three different steps in the grid-forming process. These are shown in table 4.1 below.

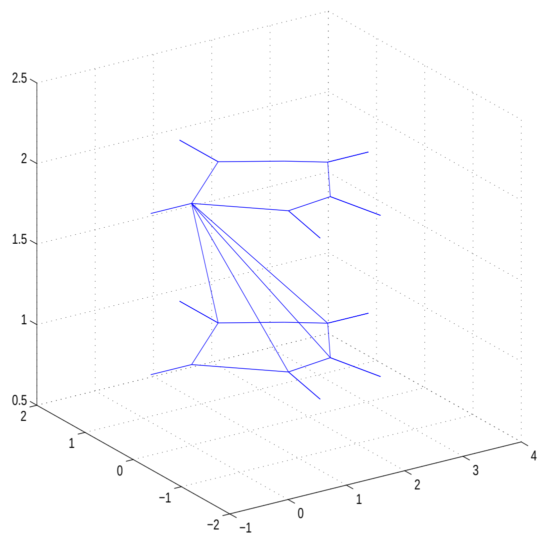


Figure 4.4: In the 3-dimensional grid the layers consist of identical Voronoi grids. Between the layers new, directed edges stretches, forming shortcuts over the Voronoi zones and representing a HARM fired at a SAM site.

The costs on the horizontal edges are kept as the edge costs proposed in section 4.2.1. On the descending edges the contribution from the engaged SAM site is set to zero, since it is supposed that it was totally knocked out by the HARM. The rest of the SAM sites are supposed to contribute to the threat just as before. Let e_h denote the set of horizontal edges on each level and e_d be the set of edges connecting an upper plane with a lower one.

Since the costs on the horizontal edges in lower grids are unaffected by the fact that some SAM sites are already shot at, the paths found in this three dimensional grid will have a higher cost than is actually the case. This is hard to avoid in the three dimensional grid, since every layer can be reached by firing at a number of different sequences of SAM sites. Storing different costs at the horizontal edges depending on which SAM sites that are engaged would require one layer for each SAM site. The number of edges and vertices would explode and any treatment of the graph would be very time consuming (see section 4.2.4). Using the unaffected costs on the edges will indicate a higher risk along the chosen path than is actually the case, and the theoretical success rate will be lower than the realistic one.

In the three dimensional grid a simple shortest path-search can be performed, using Dijkstra's algorithm, to find the shortest path from the start node in each of the layers to the end node in any of the lower layers. This search results in a shortest path and a success rate for each possible number of HARMs assigned to the mission. This set of success rates and paths can form the basis for decisions on the supplies for the mission. Using the data produced in the Dijkstra search the number of HARMs can be chosen so that the mission has an acceptable success rate and still low enough costs in HARM equipment and fuel consumption. When making this weighing between success rates and costs one must consider that the costs found are the actual costs of the path, while the success rates are lower bounds for the mission and are likely to end up with better values.

Having chosen the number of HARMs to assign to the mission a re-search can be performed to refine the path and get a better estimate of the real success rate. Again assuming that a HARM fired at a SAM site will totally remove the threat from it, the sites shot at in the three dimensional search are removed from the set of SAM

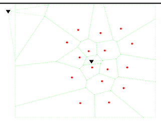
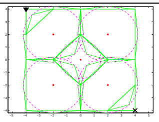
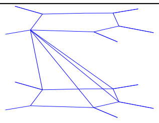
1.	Horizontal edges from the original Voronoi grid	
2.	Horizontal edges approximately following the SAM sites' ranges	
3.	Descending edges representing the engagement with a SAM site	

Table 4.1: Three kinds of edges form the three dimensional grid used to find a strategy for the SEAD UAVs.

sites. A new, two dimensional Voronoi grid is formed around the reduced set of SAM sites, and the set of edges in this grid are denoted e_r. The new search is performed on the union of the edge sets e, e_d and e_r, i.e. all the previously searched edges and the new edges added when the engaged sites were removed.

The new search will produce a path that is at least as good (has at least as high success rate) as the one found in the three dimensional search.

4.2.4 Post-processing The output from the search is a sequence of waypoints to pass on the way from start to final point. The path might however be quite coarse, especially in the outskirts of the searched area. To make the path more intuitive and not unnecessarily long, a post-processing of the path is performed.

The edges forming the path are divided into shorter segments, adding new waypoints with a certain interval. The cost to go straight from waypoint $i-1$ to waypoint $i+1$ is now compared to the cost to first pass waypoint i for every i. The point that gives the biggest cost reduction is removed, and the shortcut is now part of the planned path. New points are removed until no more savings are possible by removing waypoints (see fig 4.5)

This iteration gives a straighter and shorter trajectory and is computationally efficient since at most two new costs has to be computed in every step.

Computational Complexity In a Voronoi diagram constructed around $n \geq 3$ sites, the number of vertices, v, is at most $2n-5$ and the number of edges, e, is at most $3n-6$ [7]. Using an efficient algorithm, such as Fortune's algorithm, the construction of the diagram can be done in $\mathcal{O}(n \log n)$ time. When the additional edges following the borders of the areas covered by the SAM sites' radars are added, the number of possible trajectories is increased. The number of edges is however still kept at $\mathcal{O}(n)$.

In the three dimensional grid there are h levels (one for each HARM) and the number of vertices is hv. Each level still contains e edges, and they are now complemented by new edges between the levels. From every node in the uppermost of two levels stretches an edge to each node in the lower level that is part of the boundary of any common Voronoi zone. Since a node normally takes part of the boundary of three zones, and a typical zone has about five or six vertices, the number of new edges connecting a pair of levels will be about $15v$. In any case the total number of new edges will be $\mathcal{O}(hv)$. The size of the path planning problem is enlarged, but the

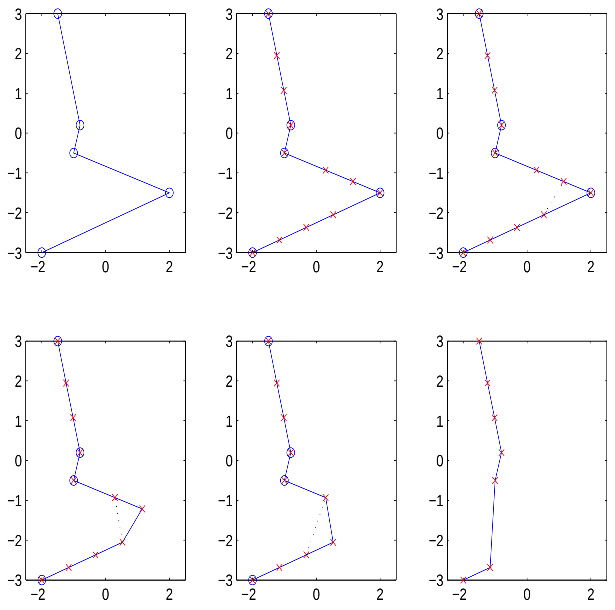


Figure 4.5: Some steps in the post-processing procedure. The first figure shows the initial coarse path. In figure two the edges are split into shorter intervals. The following three figures show a sequence of three steps in the postprocess. The dotted lines are the shortcut that improves the cost the most. The last figure shows the final post-processed path, which is both smoother and cheaper than the initial path.

complexity is still the same. The time consuming part will be the Dijkstra search, which is done in $\mathcal{O}(h^2n^2)$.

4.3 Simulations

Using the method described above, a number of simulations have been performed. The results are promising, with simple, reasonable paths, yielding high success rates.

In one series of simulations the number of HARMs is chosen automatically, based on the cost function of the different solutions in the three dimensional grid. Another simulation set has been performed where a human operator is supposed to process the different 3-D paths, and choose the number of HARMs using the cost function and her experience about the SEAD mission to guide the decision. The latter offers the possibility to compensate for aspects of the real scenario that due to simplifications were disregarded in the model.

To demonstrate these ideas to the people at the Swedish Air Force Air Combat Simulation Centre (FLSC, in Swedish) the scenario below was designed, and the output of the algorithm was fed into the distributed parallell simulation system at FLSC. Two attacking UAVs then flew the waypoint trajectory, engaged the designated SAM sites and finally destroyed the target and returned to base. The FLSC people were very interested, and since the simulation environment is ideal for further studying the interaction between operators, manned and unmanned aircraft, porting the full functionality of the algorithm seems to be a very promising future direction.

The scenario includes 16 SAM sites and a target location distributed over Gotland. The SAM sites used have two different radar ranges, 15 and 24 km. The square surrounding the available area stretches 180 km in north-south direction and 200 km in east-west direction.

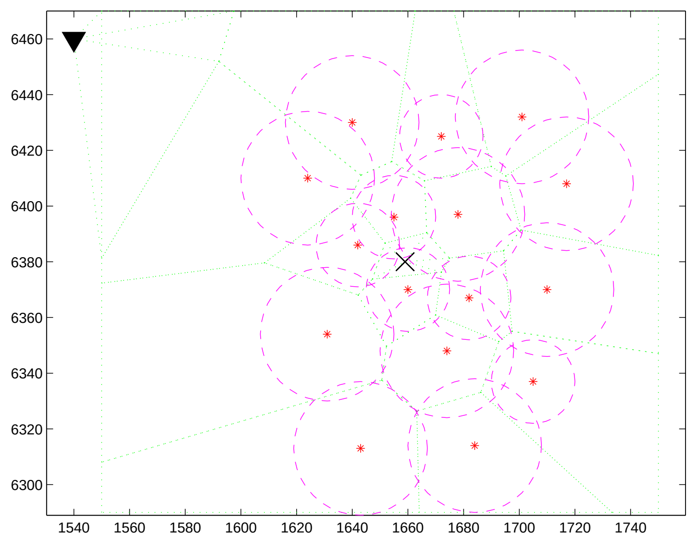


Figure 4.6: A scenario consisting of 16 SAM sites. The target is indicated by a cross and the initial position of the UAVs is indicated by the triangle in the upper left of the figure.

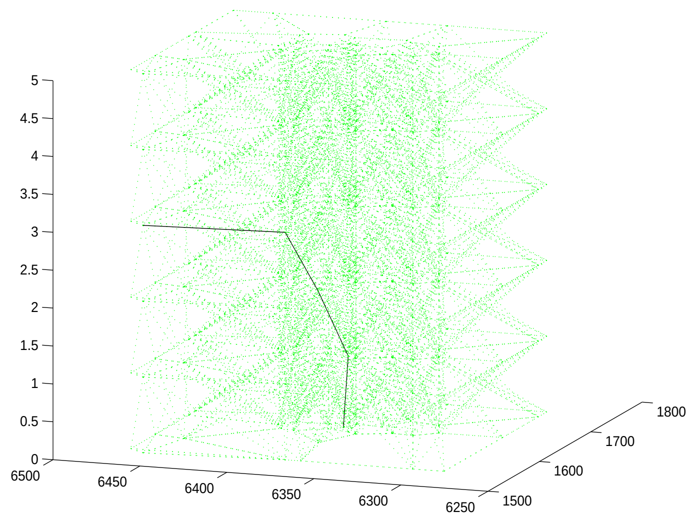


Figure 4.7: The figure shows the three dimensional grid with layers constituted of the Voronoi zone around the SAM sites. The path shown is the one that is chosen by the operator, using three HARMs.

4.4 Operator Influence on Flight Control

The proposed solution could, as is suggested in section 4.3, be integrated in a more complex, operator controlled mission planning system. Facing a SEAD mission, the human operator would run the combined path planning and target assignment algorithm to obtain possible strategies and outcome probabilities of the mission for a range of HARM and UAV configurations. Figure 4.9 gives a glance of a possible transparent interface.

Basing her decisions on this information, combined with experience of SEAD mis-

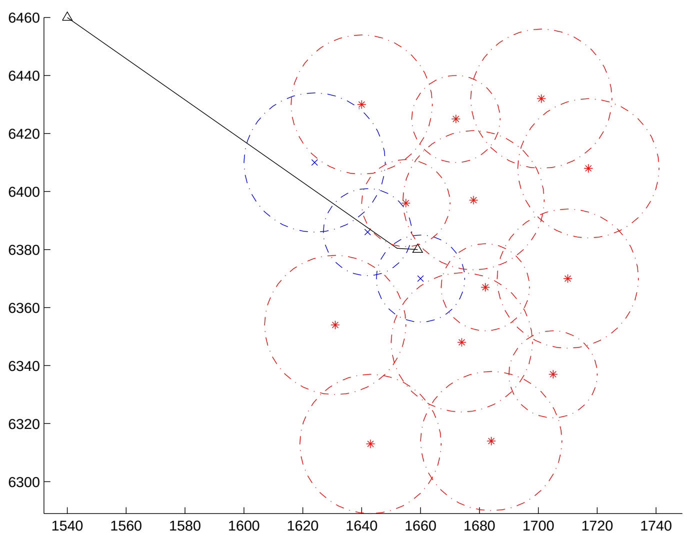
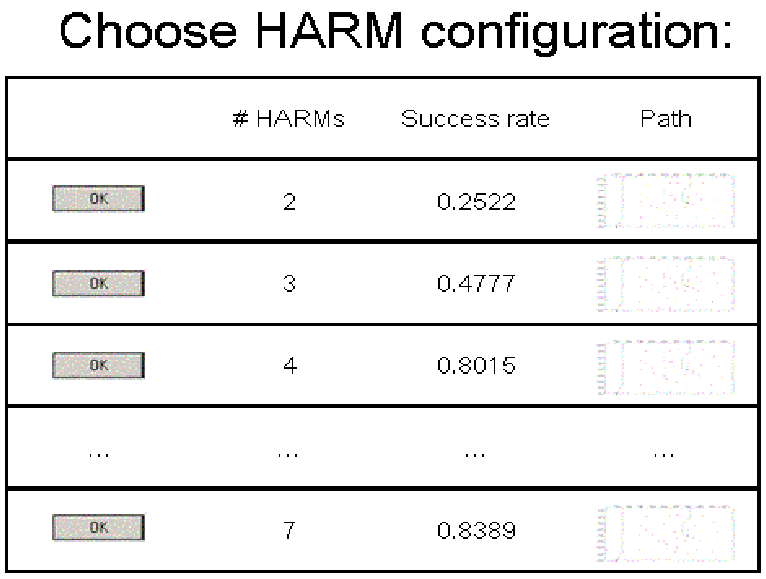


Figure 4.8: The path as it looks after re-search and post-processing. The three shot-at SAM sites are shown as crosses.



Choose HARM configuration:

	# HARMs	Success rate	Path
OK	2	0.2522	
OK	3	0.4777	
OK	4	0.8015	
...	...	...	...
OK	7	0.8389	

Figure 4.9: A possible output interface forming the decision base for the operator. The columns show success rates and paths for solutions using different number of HARMs in the mission. An operator with experience from SEAD missions may use this information to choose the UAV and HARM configuration.

sions and mission specific information (such as intelligence about the enemy's temporary weaknesses or monetary or political restrictions), the operator chooses an accurate mission setup. The risks could then be weighed against economical losses, since estimations of the risks and costs easily are obtained for a number of possible setups.

The work of the operator is not over just because the mission setup is fixed and the SEAD strike is initialized. Events that were impossible to foresee prior to the mission may occur. An a priori unknown SAM site may pop up along the path of the UAVs or a UAV may be shot down, with a reduced number of available HARMs as a consequence. At these situations the human operator uses the information obtained from the path planning and target assignment algorithm to make a good decision on whether the mission can continue or if the risks have grown so that the mission must be aborted.

Since the search through the three dimensional grid discussed in section 4.2.3 produced a shortest path and a cost as well as the success rate for every node in the grid (see section 2.2.2 about Dijkstra' algorithm), the operator can easily obtain the mission status in the case of a lost UAV. Loosing a UAV implies loosing a number of HARMs, which is represented by moving down the same number of levels in the three dimensional grid. If the data from the Dijkstras' search is saved, the operator simply checks the updated success rate, and decides whether the mission should be continued or not.

If a previously unknown SAM site, a so called pop up threat, appears on the path of the UAVs, there are two possible actions for the UAVs. Either a HARM is fired and the group of aircraft continues on the previously calculated path, or a quick turn is made, (possibly after a HARM launch) to avoid any fire from the new site. The operator may now check the success rate of continuing the mission with the number of HARMs reduced by one by looking at the cost one step down in the three dimensional grid. If the success rate goes below a certain threshold, then the operator can abort the mission, knowing that it has turned too dangerous to proceed. In cases when the pop up threat appears close to the UAV, there will be no time for a human operator to make decisions. The UAVs should then automatically fire a HARM at the new site and switch waypoint to avoid the threat. Starting from the new position, the path should be recalculated, and the operator makes the continue/abort decision based on the new situation.

4.5 Conclusions

The proposed algorithm gives good solutions to the pathplanning and target assignment problem in a simple manner. It includes the possibility for a human operator to influence the outcome of the algorithm in realtime, which is good, since the complicated reality can never be captured at a whole in a model. Alternatively, the choices of the operator could be formalised and included in the model.

The model could be improved by introducing a cost on the decending edges in the three dimensional grid, stemming from the threat posed by the shot-at site, and considering the probability that the shot missed its target or wasn't fatal. This threat should be introduced in the two dimensional grid as well, and would give a slightly more accurate picture of the threat.

Integrating the timing algorithm in chapter 3 with the pathplanning and target assignment scheme would give an even better threatening picture, using the complement of the success rate in the timing scenarios as the contribution to the threat cost from the shot at sites on the descending edges and in the two dimensional grid. This would however considerably increase the time consumption of the computations, since the dynamic programming recursion would have to be performed for every descending edge in the three dimensional grid. It is therefore reasonable to keep the problem division if the computations are suposed to be done in real time.

A possible objection to the solution proposed in this chapter is that it disregards the possibility to keep the UAVs at a low altitude, sneaking under the radars of the enemy's SAM sites. This is however easy to incorporate in the model, if desired. New layers may be introduced in the three dimensional grid, representing flying along the same paths in the plane, but at low altitude. The threat on these edges will be slightly lower, since the radar coverage of the SAM sites is avoided to a large extent. The lengths of these edges will however be increased in comparison to the same high-altidude edges, since fuel consumption is higher due to aerodynamic drag and detours forced by the topography. Even though the threat from the SAM sites is decreased at low-level flying, there are other threats that will increase. An airplane at low altitude is an easier target for Man-Portable Air Defence Systems (MANPADS) as well as for pop-up SAM sites, that may appear at a very short distance. To summarize, low altitude flying is easily captured in the proposed solution, but some disadvantages can be held against the strategy as such.

5. Engagement of Multiple Slow Moving Targets

In this chapter the final phase of the mission, where multiple UAVs engage multiple moving targets, is studied. The scenario includes a number of prioritized moving targets, possibly in a sensitive environment with civilians in the area. The task of this subproblem is to plan and execute the target assignment and the intercepting trajectories of the UAVs. These UAVs may either be the UAVs used earlier in the SEAD mission, equipped with additional air to ground ordnance, or guided sub-munitions carried and released in the target area by the strike vehicle. In the following, the term UAV will be used to denote either of these. The UAVs are parts of a communicating network, also including the command centre. This enables the vehicles to exchange and update full information about the locations of the targets and the status of the UAV fleet.

5.1 Problem Formulation

The problem is formulated as finding a winning strategy to use in the following scenario:

A group of m communicating UAVs perform a strike mission in an area where n possible moving targets are located. The nature of each target is uncertain, and given as a set of probabilities P_i for target i being any of a set of possible target kinds, e.g.

$$P_i = \left\{ \begin{array}{l} \text{p(target } j \text{ is a tank)} \\ \text{p(target } j \text{ is a truck)} \\ \text{p(target } j \text{ is a tree)} \end{array} \right. = \begin{bmatrix} p_{i,\text{tank}} \\ p_{i,\text{truck}} \\ p_{i,\text{tree}} \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0.3 \\ 0.1 \end{bmatrix}$$

The location of the suspected targets, and their probability sets, are obtained from a separate reconnaissance mission or information from a satellite. Since the strike vehicles are fast compared to the moving targets, a reasonable simplification is to consider the targets fixed. Their positions are however not entirely known, since it is unclear which of the suspected target points contains real targets (i.e. trucks or tanks), and which contains false positives (trees, civilian cars etc.).

The target points should be prioritized, and attacked by the UAVs according to the priority. The ranking of the target points should consider the importance to hit each kind of target, according to how dangerous it is, but also the probability sets of the target points, so that the effects of an incorrect target identification are kept minimal.

Some kinds of targets may need to be engaged by two or more UAVs at the same time to be disarmed. There may also be a need for so called battle damage assessment, i.e. reconnaissance by some UAVs after a strike, to investigate the effects. A solution to the problem must therefore allow simultaneous strike by a set of UAVs as well as assigning some vehicles to scouting commissions.

As the UAVs proceed in executing the strike mission, the information about the suspected targets will increase, which will change the probability sets. This happens for instance when a UAV has come close enough to the target point for its sensors to register the true nature of the target, or when one of the target points is hit by the UAVs. Due to this the proposed algorithm must include a continuous check for new information and allow re-planning as soon as new information arrives. An algorithm that is supposed to be recalculated several times during the mission execution of necessity has to be fast and simple enough to be run in real time.

In [15], the importance of decomposing the computational solution of the problem is stated. A system where all computations are performed centrally and communicated out to every UAV is undesirable for several reasons. The communication of huge amounts of information is sensitive to disturbance from the enemy and the more information that is sent through the network, the easier it will be for the enemy to locate the UAVs. Centralizing the computation at a whole also reduces the robustness of the system, since a failure of the central computer will be hazardous to the whole mission. In this mission such disadvantages are not acceptable. To truly benefit from the possibilities of a strike by multiple autonomous UAVs, these must have the possibility to re-plan their own assignments and trajectories in the case of lost communications. It is therefore preferred to decentralize the computational solution of the optimization problem as well as the trajectory planning as much as possible. The calculations are performed simultaneously on board every UAV and the only information communicated between the UAVs and the base is the status and location of the UAVs and targets. To enable this, the method must be efficient, so that it can take place in real-time on board UAVs with limited computing resources, and the individual behaviour must be reasonable as a single vehicle approach, giving acceptable results in cases of lost communication.

5.1.1 Modelling The problem described above breaks up into two main parts: the target assignment and the trajectory generation. These can be treated separately to a large extent and are both described below.

The target assignment The target assignment problem is the task of prioritzing the possible targets and select which targets should be attacked by which UAVs. The prioritization of the target points is based on the probabilities for different kinds of possible targets and on the value of hitting that kind of target. A set of weights, W, is associated with the target kinds. This set is defined separately for each mission, depending on the objectives of the mission, and on special circumstances in the area. A set of weights can be, for instance:

$$W = \left\{ \begin{array}{l} w_{\text{tank}} \\ w_{\text{truck}} \\ w_{\text{tree}} \end{array} \right\} = \left[\begin{array}{r} 20 \\ 10 \\ 0 \end{array} \right].$$

The prioritization, or cost associated with not engaging any UAV with the target, is then given by

$$c_j = \sum_{t \in \{\text{target types}\}} w_t p_{j,t}.$$

for target j.

Besides this cost of *not* assigning any UAV to the target, there is of course a cost for assigning a UAV to it. This cost is depending on the distance between the engaging UAV and the target, but also on the angle the UAV has to turn to reach the target. This is due to the increased fuel consumption at sharp turns and the limitations in where a flying vehicle can move. To reach positions inside the circle of minimal turning radius, the UAV has to take a long detour (see picture 5.1). Denoting the distance between vehicle i and target point j by d_{ij} and the angle difference by α, the cost of assigning target j to UAV i is modeled as

$$c_{ij} = d_{ij} + \frac{\alpha}{d_{ij} + \epsilon},$$

where ϵ is a design-parameter to be tuned in simulations.

Given the weights of assigning and not assigning the target points to a vehicle, the problem can be thought of as a min-cost flow problem, described in 2.2.3. The UAVs are then thought of as unit sources and the target points as unit sinks. A set of m *formation flight sinks* are added to enable the cases when targets are unassigned, and to match the number of sources and sinks n *dummy sources* are added. The set of $m+n$ unit sources and as many unit sinks now constitute a special case of the min-cost flow problem, called an assignment problem. The problem setup is illustrated in figure 5.2. A flow in the connection between UAV-source i and target-sink j represents assigning the target to the UAV, and the cost on the edge is $c_{i,j}$. Not assigning target j to any UAV is associated with a flow in the edge between any dummy-source and the target-sink, at the cost of c_j. A UAV to which no target has been assigned will continue on its pre-specified direction, flying in formation with any other unassigned UAVs and looking for a target worth attacking. This is represented by a flow from the UAV-source to any of the formation flight sinks. The cost for this transfer is a threshold value c_t, indicating at what cost a target is worth attacking. Finally the connections between dummy-sources and formation flight-sinks are used at zero cost.

The min-cost flow problem can be stated as an LP-problem, denoting the cost from source i to sink j by $f_{i,j}$ and the cost on the edges (i.e. $c_{i,j}$, c_j, c_t or 0) by $p_{i,j}$.

$$\begin{aligned} min \quad & \sum_{ij} p_{ij} f_{ij}, \\ s.t. \quad & \sum_{j} f_{ij} = 1, \\ & \sum_{i} f_{ij} = 1, \\ & f_{ij} \in \{0,1\}. \end{aligned}$$

The flow in the edges is binary, either a target is assigned to a UAV, or it is not.

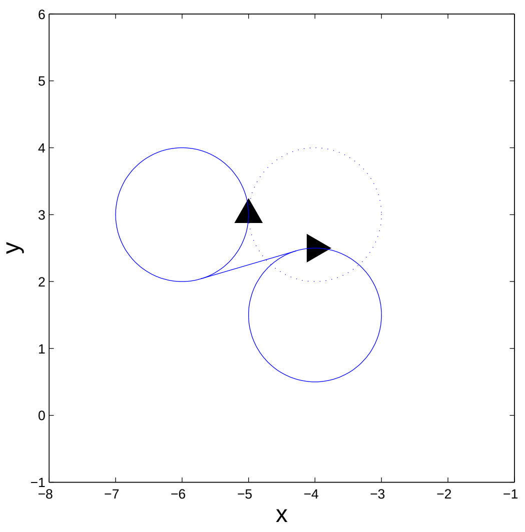


Figure 5.1: To reach a position inside the circle of minimum turning radius (dotted), the UAV has to make a detour. This makes it more expensive to reach positions at a large angle difference close to the vehicle than those further away.

The trajectory generation Once the assignment has been done, there is a need for an algorithm finding the path from the current positions of the UAVs to their assigned targets. The task of finding an accurate path from one position to another, knowing both initial and final location and attitude, can be solved in many ways. Since the length of the paths to go in this scenario are of the same order of magnitude as the minimum turning radius of the UAVs, the approach adopted in chapter 4.2.1 is inadequate. Instead, the UAVs are represented by a unicycle model, presented in section 2.4.1 and the trajectory generation turns into a control problem. How to find the proper control law to steer the UAV from the initial state $[z_1, z_2, \theta]_i$ to the desired, final state $[z_1, z_2, \theta]_f$?

5.2 Proposed Solution

To solve the problem of Engagement of Multiple Slow Moving Targets, time is discretized and the target assignment and the trajectory generation are performed iteratively in each time step. The target assignment is performed by solving the assignment problem described in section 5.1.1, either with the *alpha-beta algorithm* described in section 2.2.3 or by solving the LP-problem with simplex. A solution to the trajectory generation is proposed below.

5.2.1 Trajectory Generation Given the locations and desired orientations of the initial and final positions, two types of trajectory generators are proposed. The first assumes the vehicles to be travelling at constant speed, and finds the shortest feasible path from initial to final point with the prescribed orientations. This generator uses a shortest path algorithm developed by Sussmann, that is described in section 2.4.2.

An alternative trajectory generator finds a proper path for the vehicle by controlling a point located the distance L ahead of the UAV's centre of mass. This point

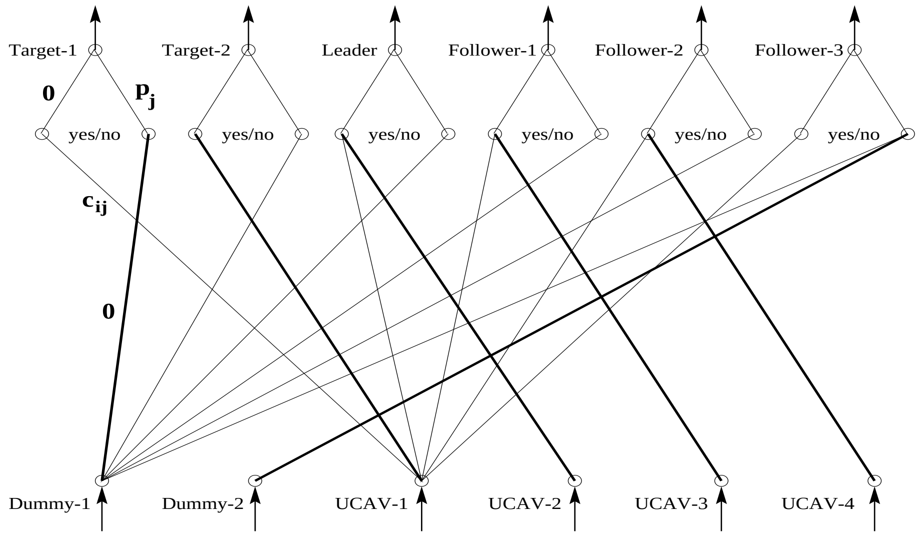


Figure 5.2: The graph of the min cost flow assignment. The edges with nonzero flow, corresponding to an assignment, are drawn as thick lines. Target-1 is assigned to Dummy-1 (i.e. unassigned), Target-2 is assigned to UCAV-1, etc. To avoid making the figure too cluttered only a subset of the zero flow edges, the ones corresponding to Dummy-1 and UCAV-1, are drawn. Note that there is a priority related cost for not assigning targets and a distance related cost for assigning them.

is denoted by (x_1, x_2) in figure 5.3 and is given by

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} z_1 + \cos\theta \\ z_2 + \sin\theta \end{bmatrix}.$$

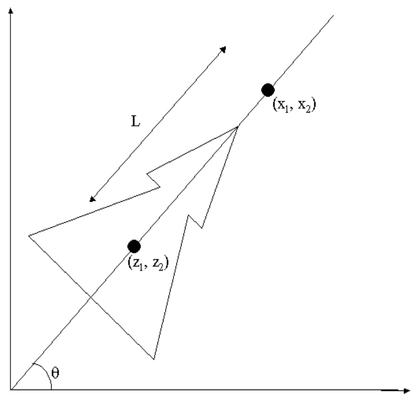


Figure 5.3: A schematic picture of a UAV represented by a unicycle model. The point (z_1, z_2) located at the center of mass and the orientation θ is the state of the system. Proper inputs to the system can be found by applying a P-controller to the point (x_1, x_2) at the distance L ahead of the aircraft.

Using a P-controller with a feed-forward term, the controls v and ω are given by

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = k \begin{bmatrix} \cos\theta & -L\ \sin\theta \\ \sin\theta & L\ \cos\theta \end{bmatrix}^{-1} \begin{bmatrix} x_{1,\text{des}} - x_1 \\ x_{2,\text{des}} - x_2 \end{bmatrix} + \begin{bmatrix} \dot{x}_{1,\text{des}} \\ \dot{x}_{2,\text{des}} \end{bmatrix},$$

where x_{des} and $\dot{x}_{\text{des}}$ are desired position and speed and k is a proportionality coefficient. Physical constraints on the UAV limits the controls; the turning angle is constrained to $-v_{\text{turn}} \leq \omega \leq v_{\text{turn}}$ and the velocity is bound to a certain positive interval $0 < v_{\text{min}} \leq v \leq v_{\text{max}}$. There is also a restriction on the acceleration of the UAV. Given the velocity in the previous time-step, v_p, the control v must stay between the limits $v_p \pm a\Delta t$, denoting the acceleration constraint by a and the length of the time-step by Δt.

Contrary to the shortest path algorithm, this controller obviously allows variable speed. However, it does not care about the orientation of the vehicle. Due to this drawback the algorithm is not suitable for generating the target intercepting trajectories. It is useful in formation flight, though, where the target-points are moving. In situations like the one shown in figure 5.4 it is crucial that the speed of the UAV is variable.

The shortest path algorithm is however also useful, when the difference in orientation of the UAV and the desired position is large. To formalize the selection of algorithm, an angle criterion is introduced. In situations when the angle difference between the orientations exceeds $\alpha > \pi/2$ the shortest path algorithm is used. Else, the P-controller gives the trajectory points to follow.

5.2.2 Algorithm The scheme followed in every time step is as follows. The target assignment problem is solved, supplying each UAV with a mission including a target point. A trajectory generator on board every vehicle takes over to compute a set of waypoints to follow. If the shortest path algorithm is used, the whole trajectory to follow is computed, while the P-controller only produces the waypoint to reach in the present time-step. If any of the paths produced by the shortest path algorithm is infeasible with respect to e.g. time or fuel constraints, the target assignment has to be redone. The cost on the infeasible path is set to ∞, and the process starts over with target assignment. In this way, the process is iterated until all the UAVs have feasible paths to follow.

Before moving on to the next time step, starting over with the iteration above, the locations of the UAVs are changed one time-step along the generated trajectories. Since the new positions may imply increased information about the targets, or even a reduced number of targets, due to a completed strike, both the information about the UAV fleet status (e.g. positions and fuel reserve) and the information about the targets (such as positions and probability sets) is updated before the new target assignment is initialized.

The proposed algorithm is schematically shown in figure 5.5.

To capture the required properties of enabling concurrent strike, scouting and battle damage assessment (BDA), some changes have to be made to the original assignment setup. Enabling scouting is easy, an extra sink representing the scouting mission is added, together with an extra dummy node to keep the balance between sources and sinks. The cost on the edge connecting the scouting node with the UAV sources is equivalent to the cost of engaging with the target in question, except that the distance from the UAV to the target, $d_{i,j}$ is reduced by the sensor range of the UAV, considering the mission fulfilled as soon as the UAV has been able to identify the target. The cost of not engaging in the scouting mission is set to the same as not engaging the target, $\sum_i p_i c_i$.

If scouting is required before the engagement of the target, the target node and one dummy are removed, to be reinserted with updated probabilities when the scouting mission is completed.

Battle damage assessment, BDA, is enabled by changing the target sink to a scouting sink as soon as the engagement with the target is completed. The node is not removed until the BDA is completed with positive outcome. In the case of negative outcome (i.e. the target was not neutralized by the strike), the target sink

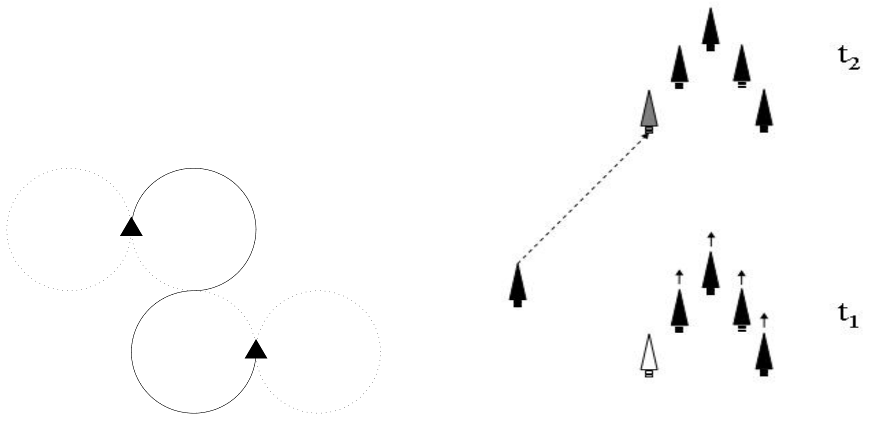


Figure 5.4: When the UAV is close enough the path found by the shortest path algorithm is not favorable. A better solution would be to allow a slight adjustment of the speed to achieve a smooth path into the formation. The left figure shows the path given by the shortest path algorithm. The right figure illustrates the case when the UAV is just slowing down a bit, slipping into the slot in the formation.

is reintroduced with updated probabilities.

When concurrent strike is required, further changes has to be done. The sink of the target that requires simultaneous engagement by multiple UAVs is no longer a unit sink, but takes a flow corresponding to the number of engaged UAVs. This change excludes the problem from the group of assignment problems. However, it still falls into the group of Hitchcock problems (see section 2.2.3). The same scheme for solving the problem (the alpha-beta algorithm) can be applied, but before the paths are generated it must be checked that the simultaneous attack requirements are fulfilled, or that the target is not assigned at all. If the target is assigned, but with too few UAVs, the assignment has to be redone, with the costs of assigning the target in question all set to ∞. If it is not worth to attack the target with a sufficient number of UAVs, then it is not worth attacking at all. The trajectory generator has to be completed with a rendez-vouz time constraint.

5.3 Simulations

A somewhat restricted version of the proposed algorithm was used in simulations of the scenario setup over *Rådmansö* in the Swedish archipelago. The used, restricted scheme did not allow simultaneous intercept or battle damage assessment, and the assignment of the UAVs to the targets or formation flight was just done once. In a fully developed version, the assignment should be performed in every timestep, to guarantee the best solution according to the available information.

The scenario setup is shown in figure 5.6 and the result of the simulation, the paths of the UAVs, can be viewed in figure 5.7. The simulation included mission planning for the UAVs until the targets were hit. Further instructions for the UAVs could be obtained with slight modifications of the code.

5.4 Conclusions

As shown in the simulations the proposed algorithm is useful in finding a strategy for UAVs involved in a strike mission. It is a time effective and simple scheme, yielding feasible trajectories, that can take place on board simple unmanned vehicles. However, some work has to be done to fully demostrate the capabilities of the method.

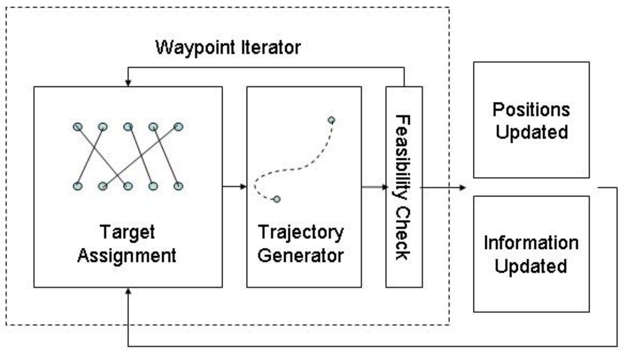


Figure 5.5: Algorithm for finding feasible strategies when facing multiple moving targets. The steps inside the waypoint iterator box are iterated in each timestep. All steps are performed locally on the UAVs, the trajectory generation and feasibility check treat only the trajectory of the UAV in question.

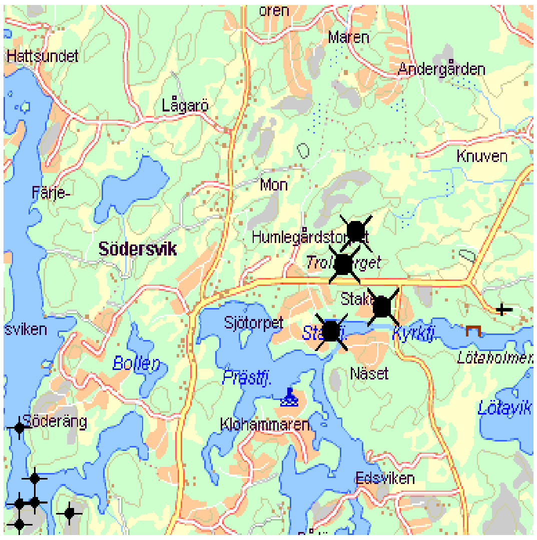


Figure 5.6: The engagement area, part of the island Rådmansö. The locations of the suspected targets are shown as big black dots and crosses to the right in the picture. The strike vehicles are shown as small dots to the left.

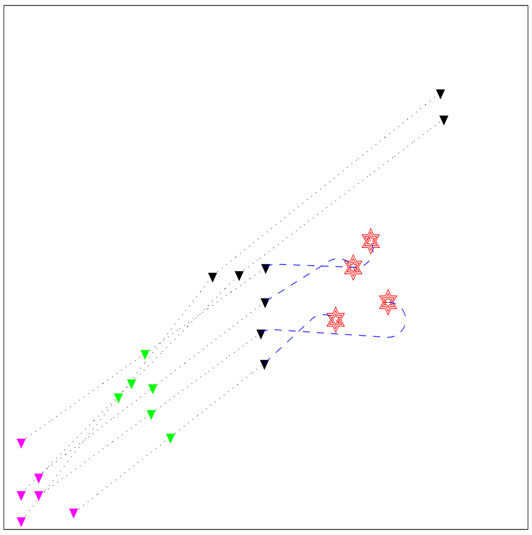


Figure 5.7: Result of the simulation. The UAVs start by ordering up in formation, and when close enough to identify the targets, four of the six UAVs attack the target points. The two remaining vehicles continue in formation to look for further targets.

6. Conclusions and Further Work

This thesis has proposed a couple of schemes that offers solutions to the two scenarios presented in chapter 1, the SEAD scenario and the Multiple Target Engagement problem. The SEAD problem is divided into two subproblems, the problem of path planning and target assignment and the question of when to fire a missile when approaching one of the assigned SAM sites.

To the first of the SEAD subproblems a solution is proposed that originates in the common path planning approach of using a Voronoi grid. The proposed solution extends this by adding new edges approximately following the borders of the SAM sites' radar ranges, thus offering possible reductions in threat. Further extensions are proposed by expanding the Voronoi grid into three dimensions, letting the number of available HARMs be represented by the third dimension. This includes the target assignment part into the path planning. In the same manner other properties, such as the flight altitude, could be introduced into the model.

The costs on the edges in the expanded grid takes into account the length of the edge as well as the threat posed on the edge by the SAM sites in a natural way. However, the threat on edges in lower layers might be overestimated, if it is influenced by SAM sites that were already shot at and defeated. Including the possibility to correct this without dramatically increasing the number of layers, and thus the computational cost, would be a desirable improvement of the model.

The solution proposed to the timing problem builds on dynamic programming. The problem is formulated as a discrete dynamic zero-sum game, where the UAV and the SAM site are the players working for opposite goals. Simulations show that the method gives reasonable results for situations with one available HARM. Further efforts should be made to implement more time efficient simulations to enable tests with multiple HARMs. There is however no reason to doubt that the results would be just as good in this case, since the method uses tools that are well known from optimization.

A reasonable improvement to the SEAD problem solution would be to merge the timing tactics into the path planning and assignment problem solution. The success rate of the zero-sum game should then be used to obtain a correct cost on the descending edges in the expanded three-dimensional grid.

Another interesting expansion of the proposed schemes is to abandon the assumption that the UAVs fly in formation. The grid should then not be sought for *one* shortest path, but a number of paths, not necessarily totally disjunct. A rendez-vouz constraint should be imposed on the arrival of the aircraft to the target area. Multiple paths are frequently treated in the literature; one approach based on Voronoi grids, that could serve as inspiration, is found in [12].

To the problem of Multiple Target Engagement this thesis proposes an iterative solution. The algorithm easily allows scouting and battle damage assessment, as well as concurrent strike, when required. Simulations are performed on a strongly simplified scenario and further studies should be performed before too generous conclusions can be drawn about the performance of the method. However, the proposed solution seems to be a natural and relevant approach to determine the behaviour of multiple UAVs performing a strike mission.

Bibliography

[1] P. Alvå, editor. *Samverkande robotar i Nätverk, Spelkort och typsituationer*. Swedish Defence Research Agency (FOI), 2003.

[2] D. Anisi. Optimal motion control of a ground vehicle. Master's thesis, Royal Institute of Technology, Stockholm, 2003.

[3] T. Basar and P. Bernhard. *H^∞-Optimal Control and Related Minimax Design Problems*. Birkhäuser, 1995.

[4] R. W. Beard, T. W. McLain, M. A. Goodrich, and E. P. Anderson. Coordinated target assignment and intercept for unmanned air vehicles. *IEEE Transactions on Robotics and Automation*, 18(6):911–922, 2002.

[5] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[6] T. Cullen and C. F. Foss, editors. *Jane's Land-Based Air Defence 1992-93*. jane's Information Group Limited, 1992.

[7] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlaag, 2000.

[8] http://www.fas.org/man/dod-101/sys/ac/ucav.htm. webpage (2004-10-15).

[9] U. Jönsson, C. Trygger, and P. Ögren. Optimal control. Lecture notes, Optimization and Systems Theory, Royal Institute of Technology, Stockholm, 2003.

[10] B. S. Lambeth. Kosovo and the continuing SEAD challenge. *Aerospace Power Journal*, 2002.

[11] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[12] F.-L. Lian and R. Murray. Cooperative task planning of multi-robot systems with temporal constraints. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, pages 2504–2509, 2003.

[13] http://www.lv6.mil.se/article.php?id=307. webpage (2004-10-15).

[14] T. W. McLain and R. W. Beard. Trajectory planning for coordinated rendezvous of unmanned air vehicles. Technical report, American Institute of Aeronautics and Astronautics, 2000.

[15] T. W. McLain, P. R. Chandler, and M. Pachter. A decomposition strategy for optimal coordination of unmanned air vehicles. In *Proceedings of the American Control Conference*, pages 369–373, Chicago, IL, June 2000.

[16] S. G. Nash and A. Sofer. *Linear and Nonlinear Programming*. The McGraw-Hill Companies, Inc., 1996.

[17] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization Algorithms and Complexity*. Dover Publications, Inc, 1998.

[18] B. Pederby and G. Werle. 75 år i luftvärnets tjänst. Stockholms Luftvärnsförening - Luftvärnets Befälsutbildningsförbund, 2002.

[19] K. Thompson. F-16 uninhabited air combat vehicles. Research Report, Air Command and Staff Collage Air University, 1998.

[20] United States Air Force Scientific Advisory Board. Report on unmanned aerial vehicles in perspective: Effects, capabilities and technologies. SAB-TR-03-01, 2003.

[21] A. von Sydow. Analys och jämförelse av SEAD-förmågorna hos JAS39 gripen samt en svensk UCAV år 2015. Master's thesis, Swedish National Defence Collage, 2003.

A. State equations

In this Appendix the state equations describing the propagation of the discrete system in chapter 3 are presented. The state of the system is described by

$$\mathbf{x} = \begin{bmatrix} s \\ sl \\ hl_1 \\ hl_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} \text{state of SAM site \{on/off\}} \\ \text{time to detonation of SAM} \\ \text{distance from HARM no 1 to SAMsite} \\ \text{distance from HARM no 2 to SAMsite} \\ \vdots \end{bmatrix}.$$

Since the UAV is bound to constant speed, the position of the UAV is related to the time t, and thus not included in the set of state variables above (in order to minimize the state space and speed up dynamic programming). The position will however be denoted $d(t)$ below. The constants used in the state equations are v_h, the HARM speed, v_s, the SAM speed and u, a constant of large magnitude used to keep track on how long a HARM has been flying without guidance. The latter will be explained more in detail in connection to the state variables hl_i below.

The control variables are

$$\mathbf{u}(t) = \begin{bmatrix} u_1 \end{bmatrix} = \begin{bmatrix} \text{fire HARM \{yes/no\}} \end{bmatrix} = \begin{bmatrix} \{1/0\} \end{bmatrix}$$

$$\mathbf{v}(t) = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \text{turn on/off radar \{on/off\}} \\ \text{fire SAM \{yes/no\}} \end{bmatrix} = \begin{bmatrix} \{0/1\} \\ \{1/0\} \end{bmatrix}.$$

Propagation of $s(t)$ The variable $s(t)$ above is the number of time instants that the SAM site has been shut down, $s(t) = 0$ if the SAM site is on, and $s(t) > 0$ if the site is off. Once the SAM site is shut down, there is a start-up delay, i.e. some time instants must pass until the radar can be turned on again. The variable propagates in time as:

$$s(t+1) = \begin{cases} v_1 & \text{if } s(t) = 0 \text{ i.e. the SAM site is on} \\ v_1 \cdot (s(t) + v_1) & \text{if } s(t) > 0, \text{ and startup delay has passed} \\ s(t) + 1 & \text{if } s(t) > 0 \text{ and startup delay has not passed.} \end{cases}$$

Propagation of $sl(t)$ The state of the SAM is given by the variable $sl(t)$. It is equal to the distance $d(t)$ if the missile is not yet launched, and holds the time to detonation if the SAM is fired. Since the SAM is lost if the SAM site is shut down, the variable $sl(t)$ returns to the state *not fired*, i.e. $sl(t) = d(t)$, if $s(t) > 0$. The evolution of $sl(t)$ is given by:

$$sl(t+1) = \begin{cases} (sl(t) - 1) & \text{if the SAM site is on and the SAM is fired,} \\ & \text{i.e. } s(t+1) = 0 \text{ and } sl(t) < d(t) \\ \lceil d(t)/v_s \rceil & \text{if the SAM is fired in this step} \\ d(t+1) & \text{if the SAM has just detoneted, i.e. } sl(t) = 0, \\ & \text{if the SAM is not fired,} \\ & \text{or if the SAM site is shut down} \end{cases}$$

Propagation of $hl_i(t)$ The distances from the n HARMs to the SAM site are given by the variables $hl_i,\ i = 1 \dots n$. The HARMs are launched in order, i.e. HARM ♯3 can be launched only when HARM ♯2 is already fired. When a HARM is not launched, its distance variable is equal to the distance variable of the UAV, $hl_i = d(t)$. The HARMs can only be fired when the SAM site is on, or was on in the previous time-step, since the UAV must have some accurate information on its position.

A HARM that is already fired when the SAM site is turned off continues on its path, however without guidance. At the moment of detonation of the HARM, the harm that it causes to the SAM site is dependent on its position when the SAM site was last on. Here the constant $u \gg hl_i$ becomes useful. In every time-step that a HARM propagates without guidance, i.e. when the SAM site is off, u is added to the value of hl_i. When the SAM site turns on again, the part of hl_i that stems from the added us is removed. This keeps track of both the actual position of the HARM and the position at the time for SAM site shut down in the same variable, hl_i.

The variables $hl_i,\ i = 1 \dots n$ propagate according to:

$$hl_i(t+1) = \begin{cases} (hl_i(t) - v_h) \bmod u & \text{if HARM is launched and SAM site is on} \\ hl_i(t) - v_h + u & \text{if HARM is launched and SAM site is off} \\ d(t+1) & \text{if HARM is not launched} \end{cases}$$

under the condition that the previous HARM is launched. If not

$$hl_i(t+1) = d(t+1).$$

This concludes the appendix.