Elói Teixeira Pereira

# Cooperative Control of Teams of Unmanned Air Vehicles


Faculdade de Engenharia da Universidade do Porto
Setembro de 2009

**Elói Teixeira Pereira**

# Cooperative Control of Teams of Unmanned Air Vehicles

FEUP

*Tese submetida à Faculdade de Engenharia da*
*Universidade do Porto para obtenção do grau de Mestre*
*em Automação, Instrumentação e Controlo*

Orientador: Fernando Lobo Pereira, Faculdade de Engenharia da Universidade do Porto
Orientador: João Sousa, Faculdade de Engenharia da Universidade do Porto
Co-orientador: Raja Sengupta, University of California at Berkeley

Faculdade de Engenharia da Universidade do Porto
Setembro de 2009

**To my parents and Teresa**

# Acknowledgments

I would like to gratefully acknowledge the supervision of Prof. João Sousa from FEUP. His advices go well beyond this thesis. He shared with me a lot of his expertise and theoretical insights, giving me a sense of research direction. I also want to thank the unconditional support of Prof. João Sousa, mainly during the time we spent in UC Berkeley, where more than an advisor I found a colleague and a friend.

I also like to express my gratitude to Lieutenant-Colonel José Morgado, who has been the leader of the Portuguese Air Force research group on Unmanned Air Vehicles. Since the first day I joined this group, Lt.Col. Morgado has been mentoring all the steps of my academic career, always with great trust and support at all levels.

I want to thank Prof. Raja Sengupta from UC Berkeley, whose thoughtful ideas greatly contributed to the success of this thesis. Moreover, Prof. Sengupta was always available to listen my research ideas and to give me valuable suggestions for the prosecution of my academic interests.

I wish to thank all my colleagues at the Science and Technology Laboratory of the Portuguese Air Force Academy. I am proud to be part of this laboratory, with an incredible work environment, where the demand for knowledge is always present in widespread areas ranging from science to history.

I want to thank all the teachers, researches, technicians, and students from AFA, FEUP, and UC Berkeley with whom I had the pleasure to work with since 2006. The list is too long to particularize all of them but I cannot leave without acknowledge Ricardo Bencatel from FEUP, whom I have been working with since the first steps of the UAV project, and Prof. Deolinda Adão from UC Berkeley for her valuable support and English lessons during my internships at UC Berkeley.

It is difficult to overstate my gratitude to my family and friends, in particular to my parents, Lourdes and Artur. They have been given me an unceasing support to all my decisions concerning my career and my life. Moreover, they always encouraged me to do my best in all aspects of life in order to be a better human being.

At last and not with least importance, I want to acknowledge Teresa, my girlfriend. I am deeply grateful for her unconditional support and love. Thank you for sharing with me all the good and bad moments of life.

# Resumo

Esta dissertação endereça um problema especifico do controlo cooperativo de equipas de Aeronaves Não-Tripuladas, nomeadamente, a re-alocação dinâmica de veículos entre equipas que executam operações concorrentes. A abordagem utilizada consiste em separar o problema em planeamento e controlo de execução. Quer o planeamento quer o controlo de execução são desenvolvidos de forma a permitir interacções de iniciativa mista, onde o operador detem alguns graus de liberdade que lhe permite ajustar o comportamento do sistema. Estas interacções permitem que o operador reaja a contingencias da missão que não foram contempladas na modelação do mundo. O planeamento para cada equipa consiste em determinar o número mínimo de veículos necessários para executar uma sequência de tarefas com uma determinada probabilidade de sucesso. As tarefas são executadas num ambiente hostil, onde os veículos enfrentam o perigo de serem destruídos. O objectivo do controlo de execução endereçado nesta dissertação é o de balancear o desempenho das equipas de forma a aumentar a robustez a diversas fontes de incerteza. O problema de controlo de execução é resolvido fazendo uso de Programação Dinâmica Estocástica.

# Abstract

In this thesis we discuss a specific aspect of the cooperative control for teams of Unmanned Air Vehicles (UAV), namely, the dynamic reallocation of vehicles among teams executing concurrent operations. Our approach consists of a nominal planning problem and an execution control problem. Both planning and execution control are developed in mixed-initiative environments, where the operator has some degrees of freedom that allows him to tune the system's behavior. These interactions gives the ability to the operator to react to contingencies of the mission that weren't taken into account in the modelation of the world. The plan for each team consists of the minimum number of vehicles needed to execute a sequence of tasks with a given probability of success. Tasks are to be executed in an adversary environment, where the vehicles face the risk of being destroyed. The goal of execution control is to balance the performance of teams in order to increase robustness to several sources of uncertainty. The execution control is implemented using the framework of Stochastic Dynamic Programming (DP).

# Contents

# List of Tables

# List of Figures

# List of Symbols and Acronyms

**Symbols**

| | |
|---|---|
| $\emptyset$ | Empty set |
| $\boldsymbol{\gamma}$ | Performance vector |
| $\pi$ | Control policy |
| $\mu(x)$ | Function that maps the state into a control action |
| $\Sigma$ | Set of discrete input variables |
| $\sigma$ | Vector with positive elements called state-relevance weights |
| $\boldsymbol{\psi}$ | Vector describing the features of the world that were not mathematically modeled |
| $\phi(\boldsymbol{\xi})$ | Partial function that models the dynamic of the exchanges in $\boldsymbol{\xi}$ |
| $\boldsymbol{\xi}$ | Operator's degrees of freedom |
| $\lambda_\gamma$, $\lambda_l$, $\lambda_t$ | Cost function weigths |
| $\Omega$ | Uncertainty set |
| $\omega$ | Uncertainty variable |
| $\boldsymbol{\mathcal{A}}$ | Transition-vehicle matrix |
| $a$ | Action Variable |
| $\boldsymbol{b}^d$ | Demanders boundary vector |
| $\boldsymbol{b}^s$ | Suppliers boundary vector |
| $\boldsymbol{b}_v$ | Binary vector that enables or prevent reallocations to happen |
| $c(x, a, \omega)$ | Cost Function |
| $c^\gamma$ | Cost due to unbalancing |
| $c^l$ | Cost due to loss of vehciels |
| $c^t$ | Cost due to transitions |
| $\boldsymbol{c}_f$ | Vector with fuel cost of transitions |
| $E$ | Event space |
| $F$, $f$ | State update functions |

| | |
|---|---|
| $f_{av}$ | Function that maps teams to their 2D average position |
| $f_c$ | Rate of fuel consumption per unit of distance |
| $\boldsymbol{f}_r$ | Vector with the fuel reserves of vehicles |
| $fuel$ | Fuel reserves |
| $\hat{f}$ | Extended update function |
| $G$ | Automaton |
| $g$ | Function that gives the changes on updating the value function |
| $H$ | Hybrid Automaton |
| $J(u)$ | Generic cost function in an optimization problem |
| $k$ | Stage index |
| $loc$ | SAM's location |
| $\mathcal{L}(G)$ | Language generated by automaton $G$ |
| $N$ | Last stage |
| $\boldsymbol{\mathcal{N}}$ | Node-arc incident matrix |
| $\boldsymbol{\mathcal{N}}_d$ | Gives the destination teams of each transition |
| $\boldsymbol{\mathcal{N}}_s$ | Gives the source teams of each transition |
| $n_f$ | Number of vehicles desired at the end of a certain task |
| $n_o$ | Number of allowed operator's interactions |
| $N_t$ | Number of Teams |
| $N_v$ | Number of vehicles |
| $N_z$ | Number of possible reallocations |
| $\mathbf{P}$ | Plan matrix |
| $P_k(x_{k+1}\|x_k, a_k)$ | Transition Probability function |
| $p_a(x, x_a)$ | Aggregation Probability Function |
| $p_d(x_a, x)$ | Disaggregation Probability Function |
| $pos$ | Vehicle's position |
| $p_{kill}$ | Probability of kill |
| $P_f$ | Desired probability of success |
| $p_s(i, j, l)$ | Probability of success of vehicle $j$ transit from team $i$ to team $l$ |
| $Q$ | Set of discrete state variables |
| $R$ | Reachable set |
| $S$ | Set of SAMs |

| | |
|---|---|
| $s_i$ | SAM $i$ of set $S$ |
| $SID$ | SAM identification |
| $T$ | Set of teams |
| $t_a$ | Time deadline for an attack |
| $T_d$ | Set of demanders teams |
| $T_s$ | Set of suppliers teams |
| $team_i$ | Team $i$ of set $T$ |
| $tasks$ | Set of tasks |
| $task_i$ | Task $i$ of set $tasks$ |
| $T_o$ | Maximum time allowed between asking for replan and accept the new plan |
| $T_t$ | Maximum time allowed for transitions |
| $u$ | Generic decision variable in a optimization problem |
| $V$ | Set of vehicles |
| $v_i$ | Vehicle $i$ of set $V$ |
| $VID$ | Vehicle Identification |
| $\mathcal{V}$ | Value Function |
| $\mathcal{V}_g$ | Greedy value function |
| $\mathbf{W}$ | Uncertainty matrix |
| $world$ | Set of world characteristics |
| $X$ | State space |
| $\|X\|$ | State space cardinality |
| $x$ | State space |
| $x_0$ | Initial state |
| $X_m$ | Set of marked states |
| $\mathbf{Z}$ | Allocation matrix |
| $\mathbf{Z}^r$ | Allocation matrix after reallocation |

**Acronyms**

**A**

| | |
|---|---|
| AFA | Academia da Força Aérea |
| AAFL | Advanced Airship Flying Laboratory |
| AIAA | American Institute for Aeronautics and Astronautics |

| | |
|---|---|
| AAA | Anti-Aircraft Artillery |
| **B** | |
| BDA | Battle Damage Assessment |
| BLOS | Beyond-Line-Of-Sight |
| **C** | |
| C4ISTAR | C4, Intelligence, Surveillance, Target Acquisition, and Reconnaissance |
| CoE | Center of Excellence |
| C2 | Command and Control |
| C4 | Command, Control, Communications, and Computation |
| COTS | Commercial Off-The-Shelf |
| **D** | |
| DARPA | Defense Advanced Research Projects Agency |
| DES | Discrete-Event Systems |
| DNHA | Dynamic Networks of Hybrid Automata |
| DP | Dynamic Programming |
| **E** | |
| EO/IR | Electro-Optical/Infra-Red |
| ECM | Electronic Counter Measures |
| EW | Electronic Warfare |
| EUROCAE | European Organization for Civil Aviation Equipment |
| **F** | |
| FEUP | Faculdade de Engenharia da Universidade do Porto |
| FAA | Federal Aviation Administration |
| FSM | Finite State Machine |
| FINAS | Flight In Non-Segregated Air Space |
| **G** | |
| GPS | Global Positioning System |
| GMTI | Ground MTI |
| **H** | |
| HALE | High Altitude Long Endurance |
| **I** | |
| iii | independent identically distributed |

| | |
|---|---|
| IEEE | Institute of Electrical and Electronics Engineers |
| IADS | Integrated Air Defence System |
| ISR | Intelligence, Surveillance, and Reconnaissance |
| ISTAR | Intelligence, Surveillance, Target Acquisition, and Reconnaissance |
| IP | Internet Protocol |
| **J** | |
| JAPCC | Joint Air Power Competence Center |
| JAUS | Joint Unmanned Aircraft System |
| JUCAS | Joint Unmanned Combat Air System |
| **L** | |
| LIDAR | Light Detection and Ranging |
| LMI | Linear Matrix Inequality |
| LAME | Low Altitude Medium Endurance |
| **M** | |
| MDP | Markov Decision Process |
| MATLAB | Matrix Laboratory |
| MALE | Medium Altitude Long Endurance |
| MAV | Micro-UAV |
| MICA | Mixed Initiative Control of Automa-teams |
| MTI | Moving Target Indicator |
| MOMDP | Multi-Objective Markov Decision Problem |
| **N** | |
| NASA | National Aeronautics and Space Administration |
| NAS | National Airspace System |
| NCO | Network Centric Operations |
| NATO | North Atlantic Treaty Organization |
| **R** | |
| RCS | Radar Cross Section |
| **S** | |
| S&R | Search and Rescue |
| SDP | Semi-Definite Programming |
| STANAG | Standard Agreement |

| | |
|---|---|
| SM | State Machine |
| SEAD | Strike/Suppression of Enemy Air Defenses |
| SAM | Surface-to-Air Missile |
| SAR | Syntetic Aperture Radar |
| **T** | |
| TUAV | Tactical UAV |
| TETRA | Terrestrial Trunked Radio |
| 3G | Third-Generation Cell-Phone Technology |
| TCAS | Traffic Collision Avoidance System |
| **U** | |
| USDoD | United States Department of Defense |
| USD | United States Dollars |
| UAV | Unmanned Air Vehicle |
| UAS | Unmanned Aircraft System |
| UCAV | Unmanned Combat Air Vehicle |
| UGV | Unmanned Ground Vehicle |
| USV | Unmanned Surface Vehicle |
| UUV | Unmanned Underwater Vehicle |
| **V** | |
| VTOL | Vertical Take-off and Landing |
| **W** | |
| WIFI | Wireless Fidelity |
| WWII | World War II |
| WIMAX | Worldwide Interoperability for Microwave Access |

# Chapter 1

# Introduction

In this thesis, we consider one problem of cooperative control of teams of Unmanned Air Vehicles (UAVs) with mixed-initiative interactions. The field of cooperative control of UAVs is an open research area with several exciting problems to be solved. Such problems have been receiving significant attention from the scientific community, mainly in the areas of control, communications and computer science. Moreover, this interdisciplinary nature imposes difficulties on reaching a systematic or largely accepted theoretical framework to model, design and to analyze UAVs (or other kind of agents) performing cooperative tasks.

This emergent research field is also raising interest outside the scientific community. The defense community is actively involved mainly because cooperative control of teams of UAVs allows taking human crews out of combat fields in an efficient and effective way. Security and civil protection are also interested in such systems mainly for applications where the human presence is highly undesirable like operation in contaminated areas (e.g. damage assessment in nuclear disasters), and operation during large periods of time (e.g. surveillance of critical facilities).

Tasks of cooperative control are quite complex. A common way to deal with such complexity is to hierarchically decompose the overall task into several simple tasks, which are organized into levels of control. The first level of control is responsible for the autonomous flight of each UAV. This typically involves distributed controllers that implement a set of simple maneuvers, such as flying to a certain waypoint. The level above the autonomous flight is the cooperative control of UAVs. The flight formation is

the most widely studied task of such level. In a formation, the vehicles maneuvers are coordinated in order to perform a certain collaborative task, for example, an air refueling procedure or a cooperative forest fire surveillance. An interesting application for flight formation is replacing large and high-cost platforms by several cooperative, small, and low cost UAVs.

Our problem lies at a higher level of control where the coordination is performed among teams of UAVs. We address the problem of balancing the real-time performance of teams of UAVs by dynamically reallocating vehicles among teams. To demonstrate approach we use a case study of a Strike/Suppression of Enemy Air Defenses (SEAD) mission where teams of Combat UAVs (UCAVs) have to perform a set of tasks aiming at the suppression of a set of threats. Tasks are composed by a sequence of targets and threats to strike and the probability of success desired for the completion of the corresponding task. To perform tasks, the teams are allowed to execute three types of maneuvers:

- Fly cooperatively towards a target;
- Strike a target;
- Send vehicles to other teams.

All tasks of teams are supervised by the same Command and Control (C2) facility which is responsible to establish the initial package of vehicles that addresses each task (we designate these packages by teams) and to reallocate the UCAVs during the mission execution. The threats and the targets are elements of Integrated Air Defense Systems (IADS) such as C2 facilities, Surface-to-Air Missiles (SAMs), Anti-Aircraft Artillery (AAA), and Air Defense Radars. Some threats are known *a priori* while others may only be known during the execution of the mission. The missions are conducted in an adversarial environment and, consequently, some vehicles may be brought down or even run out of assets, such as bombs or fuel. Due to uncertainty during the execution some teams may lack vehicles while others may have them in excess. If it is feasible, the dynamic reallocation of vehicles among teams may balance the needs of teams and, thus, improve the overall performance.

We address the problem of dynamic reallocation of vehicles among teams by combin-

ing techniques from dynamic programming with mixed initiative interactions (human operators in the control loop). In this framework the operator is allowed to constraint allocations, allocate permanently UAVs to teams, or even tune some parameters involved in the dynamic programming problem, which we will see further in this thesis. The dynamic reallocation of vehicles among teams could be interpreted as a procedure which balances the performance of teams. This balancing procedure increases the robustness of the mission under uncertainty. The most relevant uncertainty sources in our SEAD mission model are the possibility of vehicles being shot down by SAMs or by unknown threats, and the possibility of SAMs being inactive or already suppressed by other forces. There are other types of uncertainties that are not specific of SEAD missions such as the weather. The uncertainties in the weather forecast could lead to unexpected fuel consumption of vehicles. Mixed-initiative interactions also provide a mean to increase the system's performance. The interactions of experienced operators in the control loop enrich the overall system with strategic insights and with information from the world that were not considered in the mathematical models. In fact, with mixed-initiative interactions, both agents of control (the computation framework that solves the mathematically modeled control problem and the human operator) are committed to do their best in order to increase the performance of the overall system. For example, in combat missions the theater of operations is constantly evolving. Thus, in unforeseen situations, the operator is allowed to intervene pro-actively to avoid undesired or unsafe states.

Our control problem is divided in two phases: a planning procedure to establish the minimum number of vehicles that each team should have at each stage; and an execution control that dynamically reallocates vehicles among teams. The planning procedure and the execution control are designed with mixed-initiative interactions. The problem is modeled as a Markov Decision Process (MDP) and it is solved using the Stochastic Dynamic Programming framework.

## 1.1 Previous work

In this section we briefly describe the work that has been performed in the field of UAVs by the Portuguese Air Force Academy (AFA) and the School of Engineering of the University of Porto (FEUP). AFA started in 2001 a project to design, construct and

remotely operate several fixed-wing platforms with wingspans ranging from 2 to 6 meters. In 2006, AFA and FEUP started a joint program for the development of autonomous UAVs based on the AFA airframes. Until now, we have been integrating commercial autopilots in our airframes and operating them successfully. Our UAVs are now able to perform waypoint-based missions, configurable by the operator, and autonomous take-off and landing. FEUP has been also developing a C2 framework called Neptus that aims at the control of multi-vehicle systems (air, surface and underwater vehicles). Neptus has been extensively used to control Unmanned Underwater Vehicles (UUV) and, in the Spring of 2007, was successfully tested to control a UAV. Neptus is compliant with the NATO[1] standard for UAV control systems STANAG[2] 4586, an important feature towards the interoperability of systems. Currently, we are mounting on each UAV a PC-104 stack, a computational device that will allow us to perform more complex on-board tasks such as target acquisition, following terrain features, following mobile targets, and multi-vehicle coordinated maneuvers. More about the AFA/FEUP actual and future developments on the UAV field can be found in [5] and [6].

## 1.2 Thesis overview

In this section we provide an outline of this thesis.

In chapter 2 we start with a brief overview on the use of UAVs in the military context. We give emphasis to military applications due to the affiliation of the author and because, up to now, UAVs have been massively used in this context. We provide the reasons that are constraining the use of UAVs in civil applications and also describe the future challenges facing UAV systems.

In chapter 3 we provide the mathematical background for this work. This background material is organized in order that the reader understands how to model, analyze, and control dynamic reallocations.

In chapter 4 we provide a literature review focusing the efforts of academia to solve problems related to the dynamic reallocation of vehicles among teams. We focus on

[1] Acronym for North Atlantic Treaty Organization.
[2] Acronym for Standard Agreement.

problems concerning networks of vehicles, load-balancing procedures, control hierarchies, and mixed-initiative interactions.

In chapter 5 we aim at providing a clear and concise problem statement. Our goal is to model a SEAD mission with a simple and intuitive framework that highlights the dynamic reallocation problem. First we present a matrix based framework to model the SEAD mission. Next we provide a procedure to plan SEAD missions and finally we address the execution control with mixed-initiative interactions.

In chapter 6 we describe our approach to solve the problem stated in chapter 5. First we present the control architecture based on a hierarchical decomposition. Next we develop an algorithm based on the Dynamic Programming (DP) framework to solve the execution control problem. Finally, we model the operator's interactions, describe the conditions under which he is allowed to intervene, and how the system provides aids to his aid his decision process.

In chapter 7 we present the simulation framework developed during this thesis. We focus on the inputs and outputs of the main software modules and we finish the chapter with a simulation example.

Finally, in chapter 8 we state the most relevant conclusions of this work and we present some ideas for further work.

## 1.3 Main contributions

The main contribution of this thesis is the development of a new framework to perform dynamic reallocations of UAVs among teams. This framework considers heterogeneous vehicles and controllers implemented with mixed-initiative interactions, allowing the operator to intervene during the planning and execution control phases. The main findings of this thesis were presented in the conference "Unmanned... Unlimited" organized by the American Institute for Aeronautics and Astronautics (AIAA) at Seattle, 6-9 April 2009 [7].

# Chapter 2

# Overview of Developments in UAV Systems

We provide background information on developments in UAV systems (UAS) with special emphasis on UAS for military applications. We start with some historical remarks followed by a taxonomy for UAVs. We also discuss in more detail the current and future roles of UAS in military operations. We conclude presenting current technical challenges emphasizing certification and inter-operability, which are crucial for the deployment of civilian applications.

## 2.1 UAV background

The interest on unmanned aircrafts started during the early days of aviation. At that time, UAVs were viewed as a bombing weapon with interesting autonomy and range capabilities. Several UAV projects were launched during the World War I and World War II with the purpose of developing "flying bombs" (what we actually designate as "missiles"[1]). The German V-1 "BuzzBomb" is known as the first successful missile. During the World War II (WWII) the V-1 was fired over London causing thousands of casualties[8].

The current spectrum of missions is quite different from the earlier days of unmanned aviation. Due to the emergence of several technology fields like sensors, computation, communication, navigation and control, modern UAVs are capable to perform wide spectrum of mission types, such as reconnaissance, surveillance, and even combat. This

[1]A missile differs on a rocket mainly on two aspects: a missile has control surfaces and a guidance and navigation payload. A rocket has just thrust capabilities.

wide spectrum of missions has been motivating the aeronautic industry to design and manufacture UAVs based on heterogeneous airframes. In fact, one can find fixed-wing airframes with wingspans with more than 20 meters like the Global Hawk or with just 30 centimeters like the Wasp. Several Vertical Take-off and Landing (VTOL) UAVs have also been successfully developed and operated (e.g. Fire Scout and Hummingbird) mainly for Naval applications. Unmanned airships such as the AAFL[2] are also starting to deploy. This heterogeneity asks for some form of classification. Next, we briefly discuss the issues of UAV classification.

## 2.2 UAV taxonomy

The military and industry communities have been developing several classification taxonomies for UAV' systems. Each organization has been developing or choosing the classification taxonomy that better fits their role of missions. Thus, we are facing a lack of a common taxonomy that leads to a correct standardization of systems.

The most common criteria for classification are derived from maximum flight altitude, operational range, and operational use. NATO adopted a simple classification dividing UAV' systems in Tactical UAV (TUAV), Medium Altitude Long Endurance (MALE), and High Altitude Long Endurance (HALE). A TUAV tipically has a flight altitude below $15,000\,\text{ft}$, a MALE between $15,000 - 45,000\,\text{ft}$, and a HALE above $45,000\,\text{ft}$ [9]. It is also usual to find some adaptations of this classification such as Low Altitude Medium Endurance (LAME).

The United States Department of Defense (USDoD) Unmanned Systems Roadmap 2007-2032 [10] presents two classification systems. The first comes from the Joint Unmanned Aircraft System (JAUS) Center of Excellence (CoE) classification system. The NAS classification system is slightly broader than JAUS classification. NAS divides UAVs in six levels (level 0 - level 5).

In [11] a more detailed classification including UCAVs, decoys, and stratospheric UAVs is presented. But, for the purpose of this thesis, it is more convenient to stick to a simple classification so, in what follows, we adopt primarily the NATO and JAUS classifications

[2]Acronym for Advanced Airship Flying Laboratory.

<table>
<tr><th>JUAS Categories</th><th>Operational Altitude (ft)</th><th>Launch Method</th><th>Weight (lbs)</th><th>Airspeed (kts)</th><th>Endurance (hrs)</th><th>Radius (nm)</th></tr>
<tr><td>T1 - Tactical 1 user: Special Operations Forces (SOF)</td><td>$\leq 1000$</td><td>Hand Launched</td><td>$\geq 20$</td><td>$\leq 60$</td><td>$< 4$</td><td>$< 10$</td></tr>
<tr><td>T2 - Tactical 2 user: battallion/ brigade</td><td>$\leq 5000$</td><td>Mobile Lauched</td><td>20 -450</td><td>$\leq 100$</td><td>$< 24$</td><td>$< 100$</td></tr>
<tr><td>T3 - Tactical 3 users: Division/ corps</td><td>$\leq 10000$</td><td>Conventional or VTOL</td><td>450 -5000</td><td>$\leq 250$</td><td rowspan="3">$< 36$</td><td rowspan="2">$< 2000$</td></tr>
<tr><td>O - Operational user: Joint Task Force</td><td>$\leq 40000$</td><td rowspan="2">Conventional</td><td>$\leq 15000$</td><td rowspan="2">$> 250$</td></tr>
<tr><td>S - Strategic user: National</td><td>$> 40000$</td><td>$> 15000$</td><td>Theater Wide</td></tr>
</table>

Table 2.1: JUAS classification system.

systems.

## 2.3 Military operations

The need of a taxonomy for UAV missions is also an important issue towards higher levels of interoperability[3]. This fact motivated [12] to propose a taxonomy for UAV missions based on a classification of the types of the operator interventions across the whole spectrum of missions. The main goal of [12] is to enumerate for each type of mission, the functional and information requirements that can serve as guidelines in the

[3]Interoperability will be discussed in section 2.4.5.

design of operators interfaces.

The NATO Joint Air Power Competence Center (JAPCC) provides a broader classification in [9]. JAPCC divides missions in four main groups, each of them with specific items. This classification is not exclusive for UAVs. It is a generic classification that could be applicable to other types of military systems (manned or unmanned). Thus, it is suitable to be used on Network Centric Operations (NCO) where heterogeneous systems exchange information and commands in a wide network in order to achieve a competitive warfighting advantage. Thus, a UAV's mission is classified as one piece of the large NCO puzzle rather than be classified independently. Table 2.2 presents the main groups and some mission' items with higher priority for NATO JAPCC.

| Mission Group | Mission | Priority |
|---|---|---|
| C4ISTAR | Reconnaissance and Maritme Patrol<br>Surveillance and Battle Patrol<br>Precise Target Location and Designation<br>Signals Intelligence<br>Communications and Data Relay | High<br>High<br>High<br>Medium<br>Low |
| Combat | SEAD<br>Overwatch<br>Electronic Warfare | Low<br>Low<br>Low |
| Combat Support | Tactical Aerial Delivery and Resupply<br>Combat Search and Rescue<br>Aerial Refueling | not specified<br>not specified<br>not needed |
| Combat Services/ civil missions | Couter Narcotics<br>Border Control and Monitoring | not specified<br>not specified |

Table 2.2: NATO-JAPCC Missions

Table 2.3 presents the US DoD top-priority missions for UAV' systems (source: [1]). This table gives some examples of manned aircrafts and UAV' systems that perform (or are planned to perform) those missions. The missions specified by US DoD mainly fit under

| Mission | Conventional Aircraft | UAS |
|---|---|---|
| Intelligence, Surveillance and Reconnaissance (ISR) | U-2, SR-71, ... | Pioneer, Global Hawk, Predator,... |
| Strike/Suppression of Enemy Air Defence (SEAD) | EA-6B | Hunter, J-UCAS |
| Electronic Attack (EA) | EC-130H Compass Call | J-UCAS |
| Network node/ Communications Relay | ABCCC | Global Hawk |
| Aerial Delivery/Rssupply | C-130 | Buster |

Table 2.3: Priority US DoD missions for UASs

the groups presented by JAPCC. However, there are some dissonances in prioritizing missions. The C4ISTAR[4] group, which includes ISR[5] missions, is the only group which gathers some consensus. On the other hand, combat missions appear to be the group of missions where priorities appear to differ significantly. Note that, besides being autonomous air vehicles, missiles are not included in UAV classifications. To make clear the difference between a UAV and a missile we can state that a UAV should have the capability to be recovered after the mission. Generally, a missile doesn't have such capability.

The US DoD claims that UAS have already achieved a stage of maturity where the main missions have already been identified. Thus, industries and academia shouldn't be seeking niches for new missions. All the efforts should be focused on achieving higher levels of reliability and efficiency of the top-priority missions.

Let us now briefly describe C4ISTAR-ISR missions, which are crucial for both NATO and US DoD, and Combat-SEAD which is an exclusive DoD priority.

[4]Acronym for Command, Control, Communications, Computation, Intelligence, Surveillance, Target Acquisition, and Reconnaissance.

[5]Acronym for Intelligence, Surveillance, and Reconnaissance.

### 2.3.1 C4ISTAR/ISR

In [1], the US DoD divides the ISR mission into three segments: "standoff", "over flight" and "denied". *Standoff* ISR missions are almost exclusively used in peace time over land and coast lines. This mission is always accomplished with full authorization of the local airspace authority. The main requirements for the UAV systems performing *standoff* missions are long endurance, in order to be able to accomplish persistent missions, and a high altitude if large footprints are needed. The RQ-4 Global Hawk is a system suitable to perform such missions. However, platforms designed to operate at high altitudes have expensive airframes, require high resolutions from the onboard sensors and expensive long haul communications. Thus, a trade-off has to be done between using a single high altitude platform (such as the Global Hawk) or using teams of inexpensive low altitude platforms (e.g. Scan Eagles) operating in a network environment.

*Over flight* missions are executed with or without the authorization of the airspace authorities but with a low risk. There are no additional requirements for this kind of mission when compared to the *standoff* missions.

*Denied* missions are performed without the authorization of the foreign airspace and the risk is high. Until recently, *denied* missions have been accomplished with the help of space technology, namely satellites. But the use of satellites has the inconvenient of being easily detected and jammed due to the predictability of their paths. Some manned aircrafts have been performing this kind of mission (U-2, SR-71); however having a human crew on board has the inconvenience of the risk of losing the crew and the lack of persistence related to human physiological limitations. UAVs can perform stealth missions in a persistent way without the additional constraint of having a human on board. The requirements for this type of missions are high levels of survivability. This objective could be reached by using armed platforms and with low radar signatures[6].

UAVs have been widely used in operational theaters performing C4ISTAR/ISR missions. For example, the Global Hawk and the Predator have been used in the Iraq and Afghanistan conflicts.

[6]The radar signature of a target is commonly known as Radar Cross Section (RCS)

### 2.3.2 Combat - SEAD

The interest on having UAVs performing combat missions was motivated by some successful trial missions in the Afghanistan and Irak war theaters with reconnaissance UAVs like MQ-1 Predator equipped with lightweight weapons. The success of such missions motivated the Predator program (MQ-9) that will provide more weapons capabilities in order to perform "reconnaissance strike" missions and "persistent strike" missions. Moreover, the US Navy and US Air Force launched a joint program designated as "Joint Unmanned Combat Air System" (J-UCAS) that aims from the development of UAVs tailored from the beginning to be combat vehicles. Besides persistent strike capabilities, the J-UCAS project wants to develop UCAVs with proper capabilities for SEAD missions. Besides avoiding the loss of human lives, removing humans from the platforms has the added advantage of removing physiological constraints from the design. This means that the maneuvering capabilities can be pushed to the mechanical limits of the platforms and that the shapes can be optimized for low RCS. These are important characteristics for UCAVs to perform SEAD missions in order to increase survivability and effectiveness.

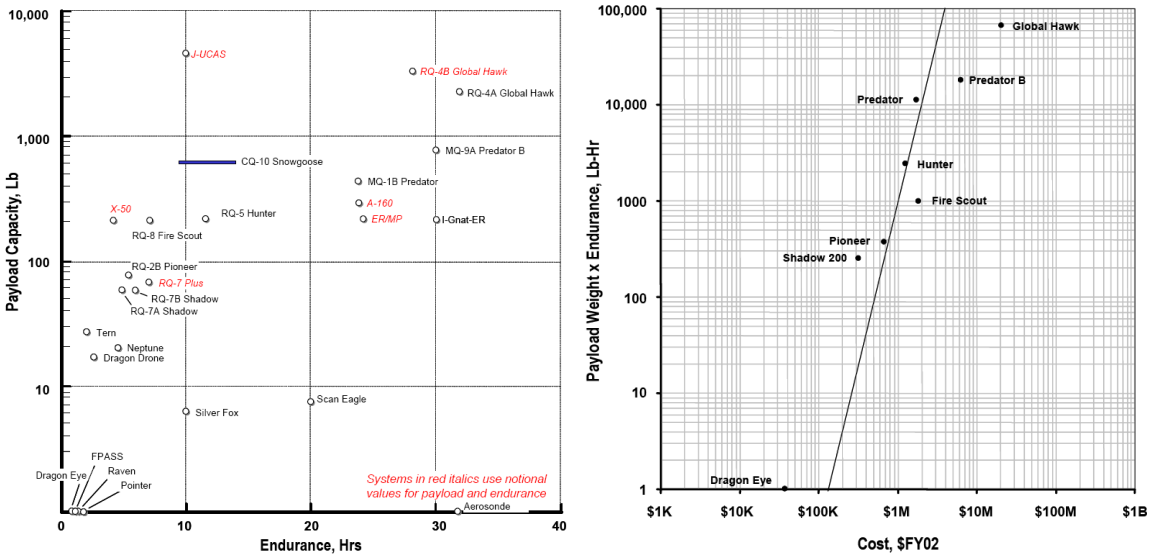
According to [1], SEAD missions are organized in two types: "pre-emptive" and "reactive". In pre-emptive missions, the path towards the target is previously cleared by other forces in the package. The reactive missions are more unpredictable, because UCAVs have to be able to react to unknown adversities; this requires high maneuverability. For example, the reaction time from detection to neutralization of a threat has to be as short as possible to reduce the exposure to risk.

### 2.3.3 Cost considerations

A key recommendation from the US DoD for future developments in UAS, which is transversal to all types of missions, concerns to the miniaturization of sensors and platforms [1]. Large platforms with high operational altitudes should only be used if it is not possible to perform the desired mission with smaller platforms with lower operational altitude. The developments should be conduced in a cost efficient way, addressing mainly small platforms with Commercial Off-The-Shelf (COTS) payloads.

Figures 2.1(a) and 2.1(b) present, respectively, the endurance and the payload capabilities

of some platforms and the cost as a function of such capabilities. Large airframes with




(a) Payload weight vs Endurance. (b) Payload weight × Endurance vs Cost.

Figure 2.1: Cost analysis in function of payload and endurance capabilities [1].

outstanding payload weight capabilities and large endurance are the obvious solution to most of the ISR missions but this comes at a high price (e.g. the cost of a Global Hawk rounds 20M USD). By using the data from Figures 2.1(a) and 2.1(b) we conclude that the price for the ScanEagle UAV is in the order of a few hundred thousand USD. Although it is not feasible to replace the payload capabilities of one Global Hawk by the combined payload capabilities of the corresponding (in terms of total cost) number of Scan Eagles, when it comes to endurance the difference is not that extreme (although the cruise velocities differ significantly). Thus, we can envision here a great opportunity in the cooperative control field (we will focus this issue in the next section).

Costs effectiveness has to be achieved without compromising the mission requirements by following the principles of systems engineering in all stages of development. During the last decades, several standards for the application of systems engineering have been developed. Some standards are transversal to all kind of systems such as the IEEE[7] Standard for Application and Management of the Systems Engineering Process [13]. Others, such as the NASA[8] Systems Engineering Handbook [14] and the US DoD Systems Engineering Fundamentals [15], are more targeted at specific applications.

[7]Acronym for Institute of Electrical and Electronics Engineers.
[8]Acronym for National Aeronautics and Space Administration.

## 2.4 Technical challenges associated to the operation of UAVs

The constant seeking for the most cost effective UASs is challenging the industry and the academia to present solutions in new fields such as Micro-UAVs (MAVs), cooperative control of teams of UAVs, and distributed data fusion applied to UAV' missions. Let us give some examples of challenges that are arising from some military missions.

### 2.4.1 Communications

The new operation theaters based on NCO concepts are demanding the exchange of large amounts of intelligence data with low latencies. ISR missions are responsible to deliver such data in a short time to Command and Control facilities to support decision making in real-time. These interactions require new communication networks with short latencies and high data rates. New network protocols have to support the integrity, availability, reliability, confidentiality, and safety of network communications and data. According to [16], the developments on military communication networks seems to point towards a IP[9] convergence of voice and data [17]. The future communication infrastructures will be based on COTS civil technologies, such as 3G, TETRA[10], WIFI[11], and WIMAX[12], adapted to the tight military requirements such as security and high reliability.

### 2.4.2 Sensors

The C4 centers needs intelligence data robust to missions' uncertainties such as weather, enemy camouflage, low target signatures, and Electronic Counter Measures (ECM) (e.g. jamming). These requirements puts emphasis on the development of complex sensors payloads for UAVs such as GMTI[13], SAR[14] [18], MTI[15] [19], LIDAR[16], Electro-Optical/Infra-Red (EOIR) systems and multi-spectral imaging systems. More about sensors technologies

[9] Acronym for Internet Protocol.
[10] Acronym for Terrestrial Trunked Radio.
[11] Acronym for Wireless Fidelity.
[12] Acronym for Worldwide Interoperability for Microwave Access.
[13] Acronym for Ground Moving Target Indicator.
[14] Acronym for Syntectic Aperture Radar.
[15] Acronym for Moving Target Indicator.
[16] Acronym for Light Detection and Ranging

for UAV can be found in [20] and [21].

### 2.4.3 Computation

The developments on sensors technologies have been increasing the number of onboard data sources. Some of this data has to be processed onboard for autonomous decision making and to expand the UAS capabilities. This requires significant onboard computational capabilities. Powerful onboard computation can perform real-time data processing, providing several interesting improvements on Intelligence, Surveillance, Target Acquisition, and Reconnaissance (ISTAR) missions. Onboard computation can provide: autonomous tracking of moving targets, following artificial and natural terrain features (e.g. routes and rivers), navigation in environments with denied or mitigated GPS[17] signals (such as in urban environments or in Electronic Warfare (EW) theaters). More about these new UAS capabilities can be found in [22], [23], [24], and [25].

Other emerging area is distributed data fusion ([26], [27], and [28]) where several UAVs take advantage of their distributed sensor and computational power to improve the performance of local data fusion algorithms with data coming from multiple sources. Several research challenges arise in this area due to communication constraints, collision avoidance, and heterogeneity of vehicles payloads.

### 2.4.4 Cooperative Control

The Cooperative control of teams of UAVs opens new possibilities for several missions, particularly for C4ISTAR missions. This is because of several factors: 1) distributed sensing by a networked team of UAVS; 2) distributed data fusion as mentioned above.

Cooperative control can also be used in persistent missions. Consider the example of a team of UAVs performing a persistent surveillance mission. Each UAV in the team is committed to survey a certain area. If one UAV runs out of fuel, then the team reconfigures itself in order to cover all the area of interest.

Combat missions also pose several interesting problems to be solved by cooperative con-

[17] Acronym for Global Positioning System.

trol. The MICA[18] program from DARPA[19] is a good example of research on cooperative networks of UAVs for combat missions. MICA aimed at the development of theories and algorithms for hierarchical control of semi-autonomous teams of UAVs performing SEAD missions. In [29], planning and execution control are developed considering experienced operators in the control loop (mixed-initiative). This is crucial in missions with high levels of uncertainty because it is impossible to model mathematically all the possible contingencies. Thus, in unforeseen situations, human operators are prompted to intervene in the system.

The cooperative control of vehicles with heterogeneous payload capabilities is also an emerging research issue. A team of UAVs with different payloads (e.g. EO, IR, and SAR) performing a ISTAR mission may enrich the intelligence data provided to the C4 centers. Missions using UAVs and others unmanned vehicles (UUV, UGV[20] and USV[21]) can be interesting to perform, for example, Beyond-Line-Of-Sight (BLOS) missions where some vehicles serves as communication relays. The US DoD gives emphasis to cooperation among several types of unmanned vehicles. This is clearly stated on the US DoD roadmap (2007-32) [10]. In comparison, the previous roadmap (2005-30) [1] was specific for aerial platforms.

### 2.4.5 Airworthiness certification and inter-operability

One of the most difficult challenges for UAS development consists of flying in non-segregated airspace shared with manned aircrafts. Due to safety of manned aircrafts crews, flying in non-segregated airspace, requires the development of strict requirements for the operation of UAS. The accomplishment of those requirements is commonly known as "Airworthiness Certification". The airworthiness certification depends on several factors such as the size of the UAVs, the type of the mission, the operational altitude and the maximum endurance. Several civil and military organizations are working, individually or in joint groups, in order to define the air worthiness requirements to certificate several types of UAVs. In 2005, the United States Federal Aviation Administration (FAA) published a memorandum (AFS-400 UAS Policy 05-01) with some technical and operational

[18] Acronym for Mixed Initiative Control of Automa-teams.
[19] Acronym for Defense Advanced Research Projects Agency.
[20] Acronym for Unmanned Ground Vehicle.
[21] Acronym for Unmanned Surface Vehicle.

requirements. The European Organization for Civil Aviation Equipment (EUROCAE) established a working group (WG73) that aims at the definition of the operational concepts for UAVs. WG73 also wants to develop capabilities to supervise and monitor all the standards, equipment requirements, and production processes regarding civil UAVs. In the military field, NATO under the joint working group "Flight In Non-Segregated Air Space" (FINAS) is working on several STANAGs in order to recommend and document NATO-wide guidelines to allow the cross-border operation of UAVs in non-segregated airspace. UAV Systems Airworthiness Requirements are defined in the STANAG 4671. More about air worthiness certification could be found in [30]. FINAS group is also addressing other problems regarding UASs such as: UAV Operator Training (STANAG 4670), application of human-interface engineering for UAV systems design (Study 4685), mission effectiveness and operational systems safety, and functional requirements for the Sense & Avoid systems.

Achieving a widely accepted Sense & Avoid system is probably the most important technology breakthroughs required towards the safe operation of UAVs in non-segregated airspace. In conventional aviation, the Traffic Collision Avoidance System (TCAS) is consensually accepted as the standard system for the avoidance of mid-air collisions. However, TCAS puts the final responsibility of the execution of the resolution maneuvers on the hands of the pilot in order that he can deal with unforeseen situations that were not detected by TCAS (e.g. presence of bird flocks and visual contact with unidentified aircrafts). The collision avoidance system for UAVs has to emulate the behavior of the pilot on such unforeseen situations.

# Chapter 3

# Background material

In this chapter we present background material for the developments of this thesis. This material covers modeling frameworks, analysis and synthesis techniques for hybrid systems, stochastic processes and dynamic programming.

## 3.1 Modeling frameworks

From a mathematical point of view, a system is a function that maps input signals (domain) to output signals (range). This function is also known as the input-output behavior of the system. Input and output signals are functions that represent the variation of a certain variable over time (continuous-time or discrete-time) or over a sequence of events. Thus, a system could be viewed as a function that transforms signals [31]. The art of finding an adequate mathematical function that emulates the behavior of a system is commonly known as "modeling". Due to the complexity and the uncertainty of real world systems, modeling is typically a difficult task and requires a careful selection of the mathematical framework. Differential and difference equations are commonly used to model, respectively, the dynamics of continuous-time and discrete-time systems. When the state space of a system can be described by a discrete set and the transitions among states occur in discrete points in time, not necessarily evenly spaced (events), we say that we have a Discrete-Event System (DES). DESs are commonly modeled by frameworks which have appealing graphical representations such as automata, graphs and Petri nets. Let us briefly describe automata based on [32].

### 3.1.1 Automata

An Automaton (also known as State Machine (SM)) is a model of a system with discrete states and triggered by events. Formally, we can represent an automaton as a 6-tuple:

$$G = (X, E, f, x_0, X_m), \tag{3.1.1}$$

where X is the state space, E is the events space, f: $X \times E \rightarrow X$ is the state update function, $x_0 \in X$ is the initial state, and $X_m \subseteq X$ is the set of marked states (also known as final states). If the state space $X$ is finite, then $G$ is a finite-state automaton or a Finite-State Machine (FSM). $G$ is said to be deterministic if $f : X \times E \rightarrow X$, i.e. given a certain state and an event, the next state is uniquely determined. If, given a state and a event, one can only know that the next state lies on a set states, then the automaton is said to be non-deterministic and $f : X \times E \rightarrow 2^X$, where $2^X$ is the power set of $X$, i.e. the set of all possible subsets of $X$. If the next state follows a probabilistic distribution then we have a Markov chain (Markov chains will be described on section 3.3). The language generated by $G$ is defined as:

$$\mathcal{L}(G) := \{s \in E^* : \hat{f}(x_0, s) \text{ is defined}\}, \tag{3.1.2}$$

where $E^*$ is the Kleen Closure of $E$[1] and $\hat{f} : X \times E^* \rightarrow X$ is the extended update function. The extended update function $\hat{f}$ is a generalization of the update function $f$ that maps a state and a string of events into another state. If $G$ is a finite-state automaton then the language recognized by $G$, $\mathcal{L}(G)$, is a regular language.

In the literature one can find two variations of an FSM that relates the automaton presented in 3.1.1 to inputs and outputs: the Moore Automaton and the Mealy Automaton (named after Edward Moore and George H. Mealy, respectively) [32]. In the Moore's automaton, there is an output function $f_o : X \rightarrow O$ that assigns each state $x \in X$ to an output $o \in O$. The inputs are basically the events that triggers the automaton. In the Mealy's automaton the transitions are of the form $guard/output$, where $guard \subseteq E$ is the input that triggers the transition and $output \subseteq O$ is an output event related with that transition. Figure 3.1 presents an example of a generic state transition diagram of a Mealy's Automaton. The arrow on the left-hand side of the Mealy's Automaton of figure

[1] If $E$ is a set of symbols, then Kleene Closure $E^*$ is the set of all strings over symbols in $E$, including the empty string.

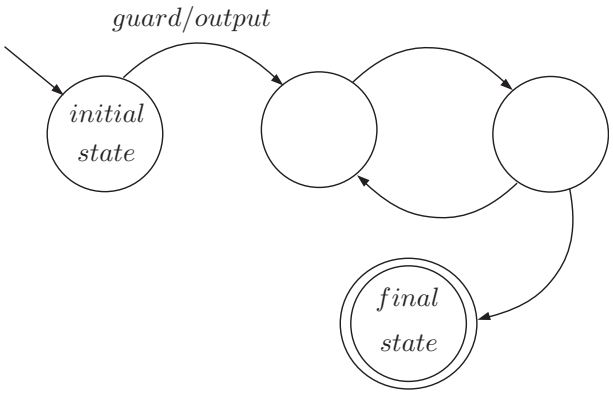



Figure 3.1: Mealy's Automaton

3.1 indicates the initial state and the final state is represented by a double circle.

Next we will present a technique suitable to model systems with both discrete and continuous states with respect to time (hybrid systems).

### 3.1.2 Hybrid systems

According to [33] there are several contexts where hybrid systems could appear: continuous time systems with phased operation (e.g. a man walking, a bouncing ball...), continuous-time systems controlled by logic (e.g. a thermostat, an UAV autopilot), and multi-agent coordination processes (e.g. the cooperative control of UAVs). For example, in the coordination of teams of UAVs, each UAV (agent) can be modeled and controlled in continuous time domain, however the assignment of UAVs to teams is modeled and controlled in a discrete-event domain[2].

When used independently, differential equations and automata lack the expressiveness required to model hybrid systems. Hybrid automata take advantage of both frameworks to accomplish such task. In figure 3.2 we present a general example of an hybrid automaton (based on [31]).

According to [34], a hybrid automaton $H$ can be formalized as the following tuple:

$$H = (Q, X, \Sigma, V, Init, f, Inv, Res) \tag{3.1.3}$$

where $Q \cup X$ is the state variables, with $Q$ a finite set of discrete states, and $X = \mathbb{R}^n$, $\Sigma$ is a finite set of discrete input variables, $V$ is the set of continuous input variables,

[2]The discrete-time systems are a particular case of a discrete-event system where the events are the clock samples.

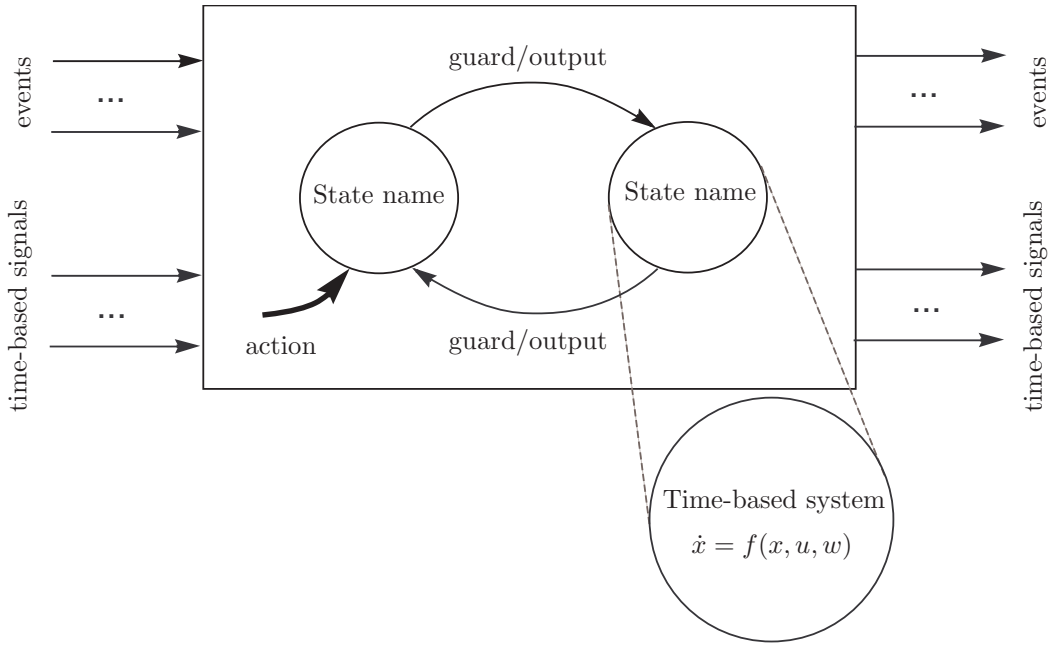



Figure 3.2: Hybrid Automaton

$Init \subseteq Q \times X$ is the set of initial states, $f : Q \times X \times V \rightarrow TX$ is a vector field describing the evolution of $x$ for each $q \in Q$, $Inv \subseteq Q \times X \times \Sigma \times V$ is an invariant that defines the combinations of states and inputs where continuous evolution is allowed, and $Res : Q \times X \times \Sigma \times V \rightarrow 2^{Q \times V}$ is a reset relation which encodes the discrete transitions of the hybrid automaton.

## 3.2 Analysis and synthesis techniques for hybrid systems

### 3.2.1 Reachability and verification

After designing a hybrid system, the designer may need to know how the state of the system evolves over time (reachability). This is crucial in order to verify if the system meet the specification and to ensure that any unsafe state is not reached by the system. A specification is a formal description of the desired behavior and properties of the system. With the specification we can describe the acceptable states, the unsafe states and the forbidden states. The process of evaluating if the model meets the specification is called verification. According to [35] there are two approaches for verification: the deductive approach and the algorithmic approach. In the deductive approach the computer and the

"human" cooperate in the verification process. The human (who has intuition about the system) suggests proof directions and the computer checks such directions for undesirable behaviors. In the algorithmic approach the computer checks the state space looking for any unsafe or forbidden state. The second approach gives more warranties that a possible unsafe state is discovered, however the number of states to evaluate could be so big that even the fastest computer would not be able to check all of them.

According to [34] some practical algorithms have been implemented for the calculation of reach sets of discrete-time systems over many thousands of possible states (an example using ellipsoids can be found in [36]), however the calculation of the reachability for continuous systems (and consequently for hybrid systems) is not a trivial task. A discussion of methods to solve reachability problems for hybrid systems is presented in [34].

### 3.2.2 Reachability of a discrete-event system

We will now give some practical considerations on reach set calculations for discrete-event systems. There are several general definitions for reach sets. The forward reach set concerns the states that can be reached from a given set of initial states with respect to time. The backward reach set concerns the states reached starting from a certain set of states and go backward to find the set of initial states. If we want to know which are the states reachable from a given set then we are concerned with forward reachability. On the other hand, if the purpose is to evaluate if the system could reach a determined unsafe state in a certain number of steps, then the backward reachability is a more efficient approach.

Suppose a deterministic finite automaton $G$ stated in 3.1.1 with a set of states $X$ and for each state $x \in X$ there is a constrained set of actions $E_x \subseteq 2^E$ ($E$ is the set of unconstrained set of events). We define $F(R) = \{x' : \forall x \in R, \exists e \in E_x, f(x,e) = x'\}$ as the function that maps all states in set $R$ to the set of possible next states, where $f : X \times E \rightarrow X$ is the state update function. Starting in the state $x_0$ the forward reachability problem could be solved by the following pseudocode:

$k = 0$
$R_k = \{x_k\}$
**repeat**

$R_{k+1} = R_k \cup F(R_k)$
$k = k + 1$
**until** $R_{k+1} = R_k$
$R = R_k$

where $R_k$ is the reach set at step $k$ and $R$ will be the final reach set. Note that $\cup$ operation in the algorithm avoids repeating the calculation for states that were previously calculated. Consider now that we have a set of unsafe states $P$ and we want to know if they are reachable. To do that we may calculate the backward reach set starting at $P$:

$k = 0$
$R_k = P$
**repeat**
$R_{k+1} = R_k \cup F^{-1}(R_k)$
$k = k + 1$
**until** $R_{k+1} = R_k$
$R = R_k$

where $F^{-1}(R) = \{x : \forall x' \in R, \exists e \in E_x, f(x, e) = x'\}$. Consider that $X_0$ is the set of possible initial states. Thus, if $R \bigcap X_0 \neq \emptyset$ some undesirable states in $P$ could be reached. Note that, in this case, the index $k$ is just a counter with respect to time.

### 3.2.3 Supervisory control

Consider that the behavior of a DES must be modified so that a set of specifications are satisfied. This task can be performed by a feedback loop of supervisory control. A supervisory controller is able to enable or disable a set of controllable events in order to satisfy the specifications. Let us consider that a DES is modeled by the automaton $G = (X, E, f, x_0, X_m)$ that recognizes the language $\mathcal{L}(G)$. The events set $E$ can be partitioned into two disjoint subsets $E = E_{uc} \cup E_c$ such that $E_{uc}$ is the set of uncontrollable events and $E_c$ is the set of controllable events, i.e. events that can be controlled or selected. The supervisor $S$ is a controller responsible to restrain the behavior of $G$ to a subset of $\mathcal{L}(G)$ in order to achieve the specifications. In other words, assuming that $S$ is able to observe all events in $E$, the controllable events $E_c$ can be dynamically enabled or disabled by $S$ [32]. Formally, the supervisor $S$ is a function that maps $\mathcal{L}(G)$ into $2^E$. According

to the notation of [32], $S$ is also denoted as the control policy and the $S(s)$ is the control action, where $s \in \mathcal{L}(G)$. Note that the domain of $S$ is not the state as in a common state feedback controller. Thus, the control action may change in different visits of the same state of $G$.

#### 3.2.3.1 Supervisory control of Hybrid Systems

The supervisory control theory can also be extended to hybrid systems. According to [2], the supervisor remains a discrete-event controller which is connected to the continuous-state plant by an interface. Such interface provides the means for the communication between the continuous plant and the DES controller. Figure 3.3 gives an example of a hybrid supervisory control system composed of a plant (e.g. a set of vehicles) that interacts with the real world with an interface that converts the continuous state variables into discrete values on a set of a finite number of elements. In practice, the discretization

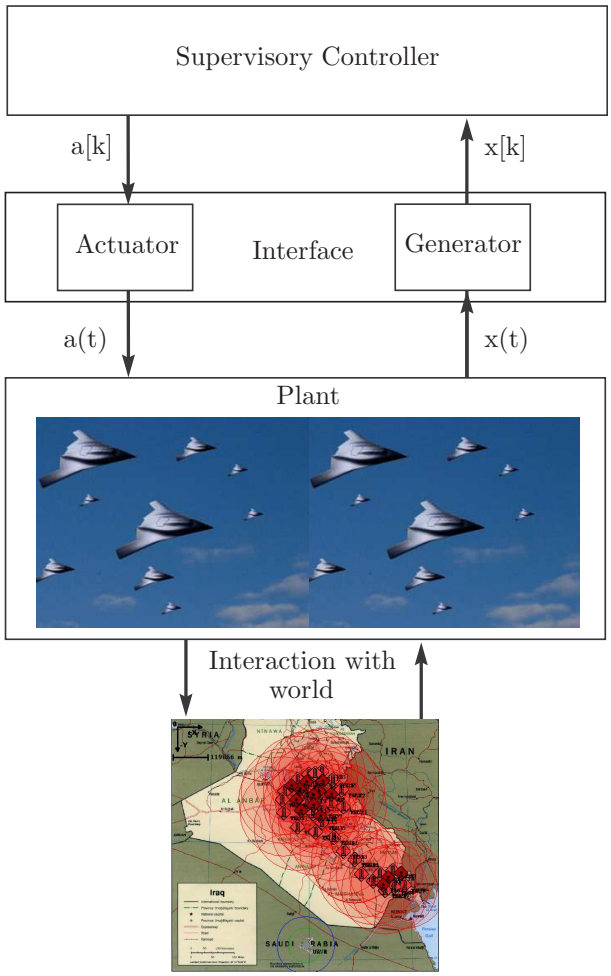



Figure 3.3: Supervisory control (adapted from [2], [3], and [4]).

of the continuous-state variables is performed by partitioning the overall continuous-state space. For example, continuous variables such as the position of vehicles, and fuel reserves are allowed to take only a certain set of discrete values.

## 3.3 Stochastic processes

In order to model non-deterministic systems we need to introduce the notion of stochastic processes. First keep in mind that a random variable is a function $X : \Omega \rightarrow \mathbb{R}$, where $\Omega$ is the sample space (e.g. when tossing a fair coin, $\Omega = \{\text{HEAD}, \text{TAIL})$. A stochastic process $X(\omega \in \Omega, t)$ is a set of random variables, defined over the same probability space, and indexed by a time variable (we use the variable $t$ and the notation $\{X(t)\}$ for continuous-time processes and the variable $k$ and the notation $\{X_k\}$ for discrete-time processes).

There are several types of stochastic processes. A simple one is the discrete-time Bernoulli process which corresponds to a sequence of independent identically distributed (iii) Bernoulli trials (such as flipping a coin). In a Bernoulli process, the future is completely independent of the past and present. A more relaxed process is the Markov process where the future is conditionally independent of the past given the present, i.e. the past history is summarized in the present state (memoryless property or Markov property). The Markov process framework is able to model a wide set of stochastic systems even in discrete or continuous-time. Let us now formalize the discrete-time stochastic processes according to [32].

Consider a discrete-time stochastic process $\{X_0, X_1, \ldots, X_n\}$. We can define the random vector $\mathbf{X}$ as:

$$\mathbf{X} = [X_0, X_1, \ldots, X_n] \tag{3.3.1}$$

where $\mathbf{x} = [x_0, x_1, \ldots, x_n]$ is a particular realization of vector $\mathbf{X}$. We are now able to define the joint cumulative density function as:

$$F_X(x_0, x_1, \ldots, x_n) = \text{Prob}[X_0 \leq x_0, X_1 \leq x_1, \ldots, X_n \leq x_n] \tag{3.3.2}$$

Let us now consider the discrete-time Markov process, also known as Markov Chain. For a Markov Chain $\{X_k\}$ the memoryless property is formally defined as:

$$\begin{aligned} &\text{Prob}[X_{k+1} = x_{k+1} | X_k = x_k, X_{k-1} = x_{k-1}, \ldots, X_0 = x_0] = \\ &\text{Prob}[X_{k+1} = x_{k+1} | X_k = x_k] = p_{ij} \end{aligned} \tag{3.3.3}$$

where $i, j \in \mathbb{N}$ represent the future and present state respectively, and $p_{ij}$ is the transition probability, i.e. the probability that the next state will be $j$ given that the present state is $i$. If the state-space is finite, the transition probabilities can be settled in a matrix form known as the transition probability matrix:

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ p_{m1} & p_{m2} & \cdots & p_{mm} \end{bmatrix}. \tag{3.3.4}$$

Markov Chains can model systems ranging from simple games, such as Tetris and Monopoly, to more complicated systems such as weather forecast, internet traffic models, and allocation procedures, such as the core problem addressed on this thesis. Next we will see how to control actions in a Markov chain.

### 3.3.1 Markov Decision Processes

In order to model a controllable Markov chain, we need to add control actions. It is also useful to associate a cost to each action. A framework that considers such control and cost features is the Markov Decision Processes (MDP) and can be formally defined as the 5-tuple:

$$\big(X, A, \Omega, P_k(x_{k+1} = i | x_k = j, a_k), c_k(x_k, a_k, \omega_k)\big) \tag{3.3.5}$$

where $X$ is the state space, $A$ is the action space, $\Omega$ is the uncertainty space, $P_k : X_k \times X_{k+1} \times A_k \to \mathbb{R}$ where $P_k(x_{k+1}|x_k, a_k)$ is the transition probability function (similar to the transition matrix but now including control actions) that gives the probability that executing action $a_k$ in state $x_k$ will lead to state $x_{k+1}$.

The optimal control of an MDP consists of choosing the optimal control actions for each state under a certain cost criterion (normally, minimize an expected cost) for the overall problem's horizon. The problem's horizon can be finite or infinite. In this work we will only consider the finite horizon case. This kind of optimization problems is commonly known as a Markov Decision Problem. Formally a Markov Decision Problem can be stated as:

$$\text{minimize} \underset{\omega_k \in \Omega_k}{\mathbf{E}} \left[ \sum_{k=1}^{N-1} c_k(x_k, \mu_k(x_k), \omega_k) + c_N(x_N) \right] \quad \forall x_k \in X_k, \forall k \in \{1, 2, \ldots, N\} \tag{3.3.6}$$

where $N$ is the problem's horizon, and $\mu_k : X_k \to A_k$ is a function indexed to the stage that maps the state into a possible control action. The control policy for the overall

horizon is designated as $\pi = \{\mu_1(x_1), \mu_2(x_2), \ldots, \mu_{N-1}(x_{N-1})\}$. Note that at stage $N$ there is no control action because basically the problem reached the end. Let us call $\mathcal{V}_\pi$ the expected cost for a certain control policy $\pi$:

$$\mathcal{V}_\pi = \underset{\omega_k \in \Omega_k}{\mathbf{E}} \left( \sum_{k=1}^{N-1} c_k(x_k, \mu(x_k), \omega_k) + c_N(x_N) \right). \tag{3.3.7}$$

We define the value function as the optimum expected cost:

$$\mathcal{V}^* = \min_\pi \mathcal{V}_\pi = \mathcal{V}_{\pi^*}, \tag{3.3.8}$$

where $\pi^* = \left\{\mu_1^*, \mu_2^* \ldots \mu_{N-1}^*\right\}$ is the optimal control policy. Next we will present an approach to solve such problems based on the Principle of Optimality.

## 3.4 Dynamic programming

In this section we will briefly describe the Dynamic Programming (DP) framework to solve an optimal control problem of a stochastic, discrete-event, finite state and finite horizon system. We give emphasis to this class of problems because it addresses the type of problems discussed in this thesis. The DP formulation can be extended to other types of problems, including, for example infinite horizon ones (see [37] and [38]). The following description is based on [39].

### 3.4.1 Principle of Optimality and Bellman recursion

According to Bellman [38] [sic.]: "An optimal policy has the property that whatever the initial state and decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision". This statement is known as the "Principle of Optimality".

Remind that equation 3.3.7 gives the expected cost starting at $k = 1$ and using a control policy $\pi$. Let us define now a "cost-to-go" function from stage $i$ to stage $N$:

$$\mathcal{V}_{\pi_i}^i = \underset{\omega_k \in \Omega_k}{\mathbf{E}} \left( \sum_{k=i}^{N-1} c_k(x_k, \mu(x_k), \omega_k) + c_N(x_N) \right). \tag{3.4.1}$$

where $\pi_i = \{\mu_i, \mu_{i+1} \ldots \mu_{N-1}\}$ is the policy for the $N - 1 - i$ steps. According to the Principle of Optimality, the remaining policy $\pi_i$ is optimal for the remaining sub-problem

$\mathcal{V}^i_{\pi_i}$. Thus, $\mathcal{V}^*_i = \min_{\pi_i} \mathcal{V}^i_{\pi_i} = \mathcal{V}^i_{\pi^*_i}$ where $\pi^*_i$ is the optimal remaining policy and $\mathcal{V}^*_i$ is the optimal corresponding value function (or cost to go). In figure 3.4 one can see a graphical explanation of the principle of optimality. The Principle of Optimality allow

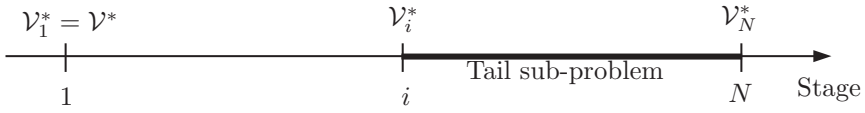



Figure 3.4: Principle of optimality.

us to calculate all sub-problems starting from the end ($k = N$) until we reach the initial stage and thus solving the problem of equation 5.4.52. This procedure can be stated as a functional equation called Bellman recursion:

$$\mathcal{V}^*_k(x_k) = \min_{\mu_k} \left( c_k(x_k, \mu_k(x_k), \omega_k) + \sum_{i=1}^{n} P_k(x_i|x_k, \mu_k(x_k)) \cdot \mathcal{V}^*_{k+1}(x_i) \right), \forall x_k \in X_k \quad (3.4.2)$$

where $n = |X_k|$ and $\mathcal{V}^*_k$ is the optimal cost-to-go function at stage $k$.

We next present a pseudo-code of a DP algorithm using the Bellman recursion:

$k = N$
$\mathcal{V}^*_N(x_N), \forall x_N \in X_N$
**repeat**
  $k = k - 1$
  $\mathcal{V}^*_k(x_k) = \min_{\mu_k} \left( c_k(x_k, \mu_k(x_k), \omega_k) + \sum_{i=1}^{n} P_k(x_i|x_k, \mu_k(x_k)) \cdot \mathcal{V}^*_{k+1}(x_i) \right), \forall x_k \in X_k$
**until** $k = 1$

We start evaluating the value function at stage $N$ and then go backwards until first stage is reached, saving the optimum control action and optimum value function for each state at each stage.

The closed-loop system is presented in figure 3.5

In literature it is common to see the Bellman's recursion defined using the operators $T_\pi$ and $T$. Let us present those operators[3]:

$$T_\pi \mathcal{V} = c_\pi + P_\pi \mathcal{V} \quad (3.4.3)$$

$$T\mathcal{V} = \min_\pi (c_\pi + P_\pi \mathcal{V}) \quad (3.4.4)$$

[3]For the sake of simplicity we will withdraw the stage index $k$.

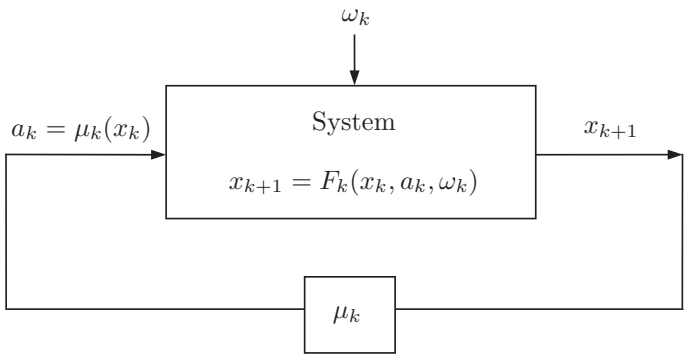



Figure 3.5: Closed-loop control.

where $P_\pi$ is the probabilities transition matrix for policy $\pi$ whose $(x, y)$ entry is the probability that the next state will be $y$ given that the current state is $x$ and the action $\mu(x)$ is according to the policy $\pi$. Thus, the Bellman's recursion could be seen, in a short hand, as the following functional equation:

$$\mathcal{V}^* = T\mathcal{V}^* \tag{3.4.5}$$

The solution of 3.4.5 is the optimal cost-to-go function $\mathcal{V}^* = \min_\pi(\mathcal{V}_\pi)$.

Solving combinatorial optimization problems using DP algorithms is usually a computationally hard task. Several relaxation techniques have been proposed to reduce the complexity while we minimize the degradation of performance. As we will see later, our dynamic reallocation control problem is a combinatorial problem and, thus, we will need to use relaxation techniques applied to the Bellman's recursion of equation 3.4.2. In fact, the high computational requirements to solve 3.4.2 is due to the huge state space. Further on this chapter we will present some relaxation techniques targeted to decrease the complexity of such problems.

### 3.4.2 Handling the uncertainties in the transition matrix

One of the input to the DP algorithm is the probability transition function $P_k(x_i|x_k, \mu_k(x_k))$. The uncertainty on this function could be handled as in [40], where a convex set of possible probabilities functions is considered and the Bellman recursion is computed for the worst case scenario. The value function in this case can be described as:

$$\mathcal{V}_k^*(x_k) = \min_{\mu_k} \max_{P_k \in \mathcal{P}_k} \left( c_k(x_k, \mu_k(x_k), \omega_k) + \sum_{i=1}^{n} P_k(x_i|x_k, \mu_k(x_k)) \cdot \mathcal{V}_{k+1}^*(x_i) \right) \tag{3.4.6}$$

where $\mathcal{P}_k$ is a convex set of $P_k$ functions at stage $k$. Further in this thesis we will present a mixed-initiative approach to deal with this problem, where instead of having a set of possible probability functions we claim that this function could be updated, with the consent of the operator, with some data provided by intelligence facilities. We choose this solution because it appears to be close to the reality of military network-centric operations. According to [41] network-centric warfare aims the implementation of collaborative worldwide information tools which enables the development of intelligence products, such as sophisticated data mining applications, that provides significantly improved access to large volumes of source data for further analysis and integration into other systems.

### 3.4.3 Aggregation approaches

The main idea concerning aggregation is to decrease the number of states. The value function of the problem is then approximated by the value function of a simpler problem, and thus reducing problem's complexity. According to [39] there are several aggregation techniques, namely: hard aggregation, soft aggregation and course grid. To understand the difference between these aggregations techniques let us first define the aggregation probability function and disaggregation probability function. The aggregation probability function, $p_a : X \times X_a \rightarrow [0, ..., 1]$, gives the probability that state $x$ is part of the aggregate state $x_a$ ($X_a$ is the set of the aggregate states). The disaggregation probability function, $p_d : X_a \times X \rightarrow [0, ..., 1]$, performs the reverse task, i.e. given an aggregate state $x_a$ what is the probability that the disaggregated state will be $x$.

#### 3.4.3.1 Hard aggregation

In hard aggregation, each original state is associated with one aggregate state. Thus, the probability function is:

$$p_a(x, x_a) = \begin{cases} 1, & \text{if } x \text{ is aggregated in } x_a \\ 0, & \text{otherwise} \end{cases} : \sum_{x_a \in X_a} p_a(x, x_a) = 1 \tag{3.4.7}$$

i.e. the original systems state $\mathcal{X}$ is just aggregated with one state aggregated state. The disaggregation probability function is uniformly distributed such that:

$$p_d(x_a, x) = \begin{cases} 1/m, & \text{if } x \text{ was aggregated in } x_a \\ 0, & \text{otherwise} \end{cases} : \sum_{x \in X} p_d(x_a, x) = 1 \tag{3.4.8}$$

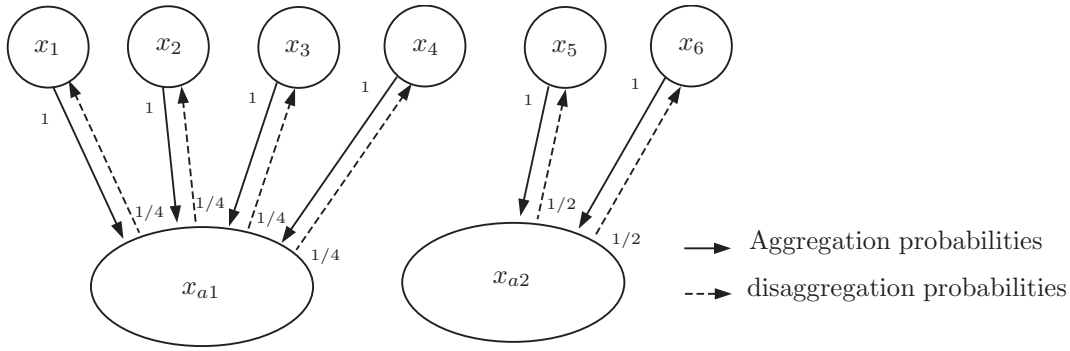



Figure 3.6: Example of hard aggregation.

where $m$ is the number of original states aggregated to $x_a$. In figure 3.6 we show a simple example of hard aggregation.

#### 3.4.3.2 Soft aggregation

In soft aggregation, each original state can be aggregate into more than one state. In this case the aggregation probability function is given by:

$$p_a(x, x_a) = \begin{cases} p_i : \sum p_i = 1, & \text{if } x \text{ is aggregated in } x_a \\ 0, & \text{otherwise} \end{cases} \tag{3.4.9}$$

The disaggregation function for the soft aggregation is the same of equation 3.4.8. Figure 3.7 depicts a soft aggregation example.

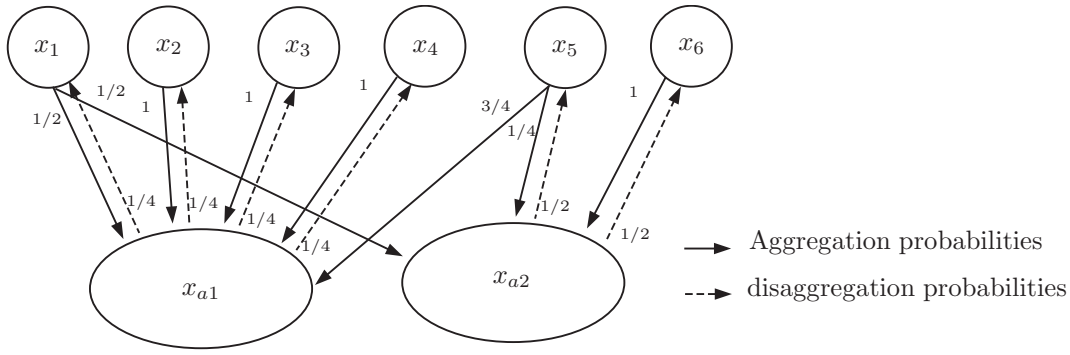



Figure 3.7: Example of soft aggregation.

#### 3.4.3.3 Coarse grid

The coarse grid is another aggregation technique that combines the ideas of soft aggregation in the sense that one original state is associated to multiple aggregate states and the idea of hard aggregation but now applied to disaggregation, i.e. each aggregate state is disaggregated into just one original state.

### 3.4.4 Relaxation techniques

Let us present briefly some techniques suitable to relax general combinatorial problems and a technique proper to relax DP problems.

#### 3.4.4.1 Convex hulls

The use of convex hulls aims at reducing the combinatorial optimization problem to a convex optimization problem. Convex optimization problems are desirable because of simpler computational requirements. In a convex problem the cost function and the inequality constraints have to be convex functions while the equality constraints have to be affine functions. The combinatorial optimization problems are non-convex because the domain of the decision variables are non-convex sets (e.g. the domain of a discrete decision vector $x : x \in \{0,1\}^n$ is a non-convex set). A simple relaxation for these problems is to consider the convex hull[4] of the non-convex set as the new domain of the problem (e.g. consider $x : x \in [0,1]^n$ instead of $x : x \in \{0,1\}^n$). After solving the relaxed problem, the optimal decision of the relaxed problem should be sampled in order to recover a decision in the original domain.

#### 3.4.4.2 SDP problem

Another technique is to relax the problem such that it could be modeled as a Semi Definite Programming (SDP). A SDP problem is defined as:

$$\text{minimize} \qquad c^T \cdot x \tag{3.4.10}$$

$$\text{subject to:} \qquad F(x) \succeq 0 \tag{3.4.11}$$

where

$$F(x) = F_0 + \sum_i x_i F_i. \tag{3.4.12}$$

The constraint $F(x) \succeq 0$ is called a Linear Matrix Inequality (LMI)[5]. A relaxation technique using SDP and applied to non-convex problems is presented in [42].

[4] A convex hull of a non-convex set is the smaller convex set that contains all the points of the original non-convex set.

[5] $F(x) \succeq 0$ means that the matrix $F(x)$ has to be positive semidefinite, i.e. all the eigenvalues of F(x) have to be nonnegative.

#### 3.4.4.3 Approximate DP

The relaxation techniques applied to DP problems are usually designated as Approximate Dynamic Programming. During the last years, the authors of [43] have been developing relaxation techniques for DP, based on approximating DP problems by Linear Programming problem. We will now briefly describe this procedure.

First consider the following problem:

$$\text{maximize} \qquad \sigma^T \cdot \mathcal{V} \tag{3.4.13}$$

$$\text{subject to:} \qquad T\mathcal{V} \geq \mathcal{V} \tag{3.4.14}$$

where $\sigma$ is a vector with positive elements called state-relevance weights. Using 3.4.5 it can be shown that any feasible $\mathcal{V}$ satisfies the inequality $\mathcal{V}^* \geq \mathcal{V}$. Thus, the unique solution of problem 3.4.13 is $\mathcal{V}^*$ for any positive values of the elements of $\sigma$. The problem 3.4.13 is non-linear due to the fact that the operator $T$ is also non-linear. However it is possible to convert the constraint of the problem 3.4.13 to a set of linear constraints:

$$c(x,a) + \sum_{i=1}^{n} P_k(x_i|x,a) \cdot \mathcal{V}(x_i) \geq \mathcal{V}(x) \qquad \forall a \in A_x \tag{3.4.15}$$

where $\mathcal{A}_\mathcal{X}$ is the set of all possible actions at state $\mathcal{X}$. We can now rewrite problem 3.4.13 as:

$$\text{maximize} \qquad \sigma^T \cdot \mathcal{V} \tag{3.4.16}$$

$$\text{subject to:} \qquad c(x,a) + \sum_{i=1}^{n} P_k(x_i|x,a) \cdot \mathcal{V}(x_i) \geq \mathcal{V}(x) \quad \forall x \in X,\, \forall a \in A_x \tag{3.4.17}$$

In [43] problem 3.4.16 is referred as the Exact LP problem. The cardinality of the state-space is enormous and, thus, the above problem has prohibitive computational requirements. To reduce the complexity, the exact LP problem is converted into an approximate LP problem. To perform that, the optimum value function is approximated by a linear combination of the basis functions $\phi_1$, $\phi_2$, ..., $\phi_J : X \to \mathbb{R}$:

$$\mathcal{V}^*(x) \approx \sum_{j=1}^{J} \phi_j(x)\, r_j \tag{3.4.18}$$

where $\mathbf{r} \in \mathbb{R}^J$ is the weight vector and $J$ is the number of basis functions. The basis can be defined as a matrix $\Phi \in \mathbb{R}^{|X| \times J}$. Thus, $\Phi\, \mathbf{r} = \sum_{j=1}^{J} \phi_j(x)\, r_j$. We are now able to

formulate the approximation of the problem 3.4.16 in the following way:

$$\text{maximize} \quad \sigma^T \Phi \mathbf{r} \tag{3.4.19}$$

$$\text{subject to:} \quad c(x,a) + \sum_{i=1}^{n} P_k(x_i|x,a) \cdot \Phi \mathbf{r}(x_i) \geq \Phi \mathbf{r}(x) \, \forall x \in X, \, \forall a \in A_x \tag{3.4.20}$$

The problem 3.4.19 is designated as the approximate LP. The number of constraints of the approximate problem could be intractable. Thus, when problem reaches this stage, another approximation could be done by sampling the constraints. The result of this new approximation is also known as the Reduced LP due to the fact that the number of constraints is reduced. More about the sampling procedure could be found in [44].

### 3.4.5 Greedy strategies/One step look ahead strategies

One way to decrease the problems' complexity is to calculate the control action only for the next step rather than for the overall horizon. These strategies are commonly known as "greedy". A greedy algorithm gives locally optimum solutions rather than global optimum solutions. However, greedy algorithms may be an efficient way to solve problems with large horizons.

Let us find a greedy version for the Bellman's recursion of equation 3.4.2. The algorithm could be stated as:

$$\mathcal{V}^*_{gk}(x_k) = \min_{\mu_k} \underset{\omega_k \in \Omega_k}{\mathbf{E}} \left( c_k(x_k, \mu(x_k), \omega_k) + c_{k+1}(x_{k+1}) \right) \tag{3.4.21}$$

where $\mathcal{V}^*_{gk}(x_k)$ is the value function for the state $x_k$ and at the stage $k$. $\mu_k$ maps the state $x_k$ into the suboptimal action. At each stage, the algorithm evaluates what is the minimum cost considering only the next stage.

# Chapter 4

# Related research

In this chapter we provide a literature review, focusing on problems concerning networks of vehicles, load-balancing procedures, control hierarchies, and mixed-initiative interactions.

## 4.1 Introduction

The problem of having several UAVs organized as teams and controlled by a single operator has been attracting attention worldwide. According to the United States Department of Defence, one of the goals for the integration of UAVs into a common airspace is [sic.] "to define appropriate conditions and requirements under which a single pilot would be allowed to control multiple airborne unmanned aircrafts simultaneously"[1]. In a military environment there are several types of missions where autonomous networked vehicles supervised by a unique operator are desirable (e.g. combat missions, surveillance, reconnaissance, and maritime patrol). However, there are several key questions that have not been properly addressed in the literature and, thus, representing open challenges to researchers: How to model and control large teams of heterogeneous vehicles? What is the best control architecture to address a certain mission? When and how should the operator intervene in order to increase performance of the system? Such questions have been addressed by researchers in areas ranging from control theory, communication, and computer science, to human factors.

In this chapter we provide a literature review focusing on the efforts of the academia to answer the above questions. More particularly, we want to conduct this review towards the particular problem addressed in this work: the dynamic allocation of vehicles among

teams.

## 4.2 Networks of vehicles

Academia is experiencing growing interest in the design and implementation of networks of autonomous vehicles, leveraged by the breakthroughs on computation, communication and sensing systems. According to Murray et al. [45] such interest will lead to the development of networked machines with a high degree of autonomy and useful for applications with high societal impact. In a network, vehicles are able to exchange information and commands among them, and their control dependencies may actually change during the execution. This distributed behavior requires a shift in conventional control theory from the usual focus on individual systems towards intricate meshes of interacting systems. Murray et al. [46] argues that the traditional control frameworks will not be able to efficiently solve control problems of large scale networked vehicles systems with strong non-linear behaviors, communications delays, and mixed discrete and continuous states. Optimization-based and Lyapunov-based control techniques (see some examples in Nilim et al. [47], Boyd et al. [48] and Freeman et al. [49]) have been proposed to address some of these problems. Bullo et al. [50] presents a new model for robotic networks that helps to formalize and analyze coordination algorithms for networked vehicles. This work also reviews distributed algorithms, graph theory and geometric optimization problems.

Researchers have focused on hybrid systems control theory to model and control systems with discrete-event and continuous-time dynamics (see Lygeros, et al. [51] and Deshpande et al. [52]). To model the dynamic interactions within individual systems researchers have been using Dynamic Networks of Hybrid Automata (DNHA) (see Deshpande et al. [53]). Sousa et al. [54] address the task allocation problem within teams of vehicles using DNHA. The allocation problem is divided in a planning procedure, based on a priori information, and an execution control. We use this approach in the architecture of our dynamic reallocation problem.

## 4.3 Load-balancing

The growth of Internet and the emergence of low-cost computing devices have been leveraging the use of distributed computing systems. This new trend requires a shift from central to distributed computation and is supported by the fact that tasks arising in a network are typically distributed. Moreover, there it is economically advantageous to use clusters of low-cost systems instead of high-cost central units. However, distributed systems only have interesting performances if the workload is fairly shared among all available computational nodes [55]. The sharing of computing power to increase performance is commonly known as *load-balancing*.

Another field where load-balancing is crucial is on multiprocessor systems. According to [56] the concentration of work on a single processor degrades the performance of that single node. Thus, the workload of computation has to be distributed uniformly over all processors. Moreover, if units of work are created dynamically during computation, rather then being already determined at compilation time, then a load-balancing algorithm has to compare the queues of processors in order to redistribute the load.

A network of vehicles shares some similarities with distributed computing systems. The vehicles could be viewed as computational devices that have to accomplish a certain set of tasks. Like in distributed computing and multiprocessors systems, the workload among teams of vehicles should be fairly distributed (commonly known in the load-balancing taxonomy as the *fairness of service*). This could be done by dynamically reallocating vehicles among teams during the execution of tasks.

The load-balancing algorithms can be divided in two main types: static and dynamic. Next we will briefly describe each type and provide some common applications.

### 4.3.1 Static load-balancing

In a static load-balancing algorithm, the decisions of sharing the load are performed based only on statistical information of the system. The main advantage of such algorithms is their simple implementation and low complexity. However, such algorithms cannot take into account the uncertainty on the workload. The static load-balancing is useful for

applications in large systems and also as a tool to help the design and the parametric adjustment of systems [57, 58].

### 4.3.2 Dynamic load-balancing

When a distributed system faces high fluctuations on the workload, the static load-balancing may not satisfy the performance's requirements. In such situations, a dynamic load-balancing algorithm may exhibit poor performance. Such algorithms dynamically balance the load according to the state of the system. However, the dynamic load-balancing algorithms are significantly more complex then static algorithms due to the fact that the decisions have to be made *on-the-fly* and based on the overall state of the system. A dynamic load-balancing algorithm can be preemptive or non-preemptive. The preemptive algorithms reassign the tasks whenever the nodes are unbalanced. On the other hand, the non-preemptive algorithms assign new incoming tasks to the suitable node at that moment. After being assigned to a node, the task is completely executed on that node. Due to the complexity of dynamic load-balancing algorithms, they have been used only on systems with homogeneous nodes and when the overhead associated to the exchange of state information during the execution is negligible [55].

## 4.4 Control architectures

When we are dealing with distributed and decentralized control applications, such as networked vehicles, the right choice of the control architecture plays an important role on system's design. According to Sastry et al. [59] it is important to correctly evaluate hierarchies and heterarchies[1] of control architectures, taking into account the centralized/decentralized behavior of decision making, the trade-off between the performance of individual agents and the optimal performance of the composite system, and the detection and management of failures. Consider a set of agents that have to cooperatively perform a task. Assume that the control scheme that leads to the optimal solution is centralized (this make sense because with a centralized version one may calculate solutions considering the whole state information). However, a centralized control architecture may be undesirable

[1] A heterarchy is a type of relationship, among agents or group of agents, that allows overlapping, multiplicity, mixed ascendancy, and divergent patterns of relation.

for several reasons: the computation of optimal solutions may be extremely complex, the demand of communications may be too high, and the reliability may be too low due to the fact that a casualty in the controller compromises the whole system. A completely decentralized solution may lead to an unacceptable performance degradation however, it may be worth to accept some performance degradation by choosing a mixed solution, where low-level controllers are decentralized (e.g. a UAV autopilot) and high-level controllers are centralized (e.g. mission supervisors). This kind of mixed solutions are also known as semiautonomous agent control [59]. Note that semiautonomous agent control is suitable to be modeled with hybrid automaton framework due to the fact that the controllers of each agent are normally continuous-time, while coordination among agents are normally discrete-time or event driven (see Pappas et al. [60] and Varaiya [29]).

Let us now consider a fully distributed team of UAVs performing a coordinated task (e.g. patrol an area). Assume that each UAV coordinates his actions with the other UAVs. Each UAV decides which action should perform based on his estimation of the state of the whole system and tries to enter into an agreement with the other UAVs. The convergence of state information and the coordination of control actions among UAVs is performed by consensus algorithms. Ren et al. [61] give an overview on the use of consensus algorithms in cooperative control for several types of dynamic systems and present some examples of application such as rendezvous, formation control, fire monitoring, and surveillance.

## 4.5 Mixed initiative interactions

The complexity of military missions is increasing. This is consequence of several factors such as the technology developments of offensive and defensive means, the constant mutation of the threats (e.g. terrorism networks), and the intervention of media and other civilians organizations directly on the war theaters. It is, thus, impossible to model the overall world behavior and create controllers that automatically react to every possible contingency. All these facts highlight the significance of having experienced operators interacting in the planning and execution control (mixed-initiative interactions), in order that, in unforeseen situations, the system may ask the operator what are the best directions [54].

Let us seek for a more formal definition of mixed-initiative. According to Allen et al. [62], [sic.] "mixed-initiative refers to a flexible interaction strategy, where each agent can contribute to the task what it does best", moreover "the agent's roles are not determined in advance, but opportunistically negotiated between them as the problem is being solved". In this same article, the authors present several examples of applications of mixed-initiative to deal with uncertainty.

As previously mentioned, Varaiya et al. [29] designed a control architecture for teams of UCAVs considering mixed-initiative interactions. In work, teams of UCAVs are supervised by only one operator who is able to interact with the system during the planning phase and the execution of the mission. Our work takes as basis the architecture designed at this work.

In the 12th International Command and Control Research and Technology Symposium (ICCRTS) - "Adapting C2 to the 21st Century", Nehme et al. [12] provided a new taxonomy addressed to the operator's interactions for missions of UAVs. In the same work, [12] emphasizes that the current air power concepts have to be readapted to include these new emergent systems, rather than pushing the development of UAVs based on obsolete concepts.

The same author presents in his Ph.D. thesis [63] a discrete-event simulation model for supervisory control of heterogeneous vehicles. In a short, this work aims at capturing the intricacies of operator-vehicle interaction during mission execution.

# Chapter 5

# Problem Statement

In this chapter we formalize the problem of dynamic reallocation of vehicles among teams executing a SEAD mission. Our goal is to provide a control framework to deal with the problem of dynamic allocation/reallocation of heterogeneous UAVs among teams that are executing operations concurrently. We start presenting a model for the SEAD mission. Next we model the attacks of teams of UCAVs to SAMs and, at last, we present the statement for the execution control of the reallocation problem with mixed-initiative interactions.

## 5.1 Introduction

As previously mentioned, our case study is based on a problem of Strike/Suppression of Enemy Air Defenses mission, where teams of Unmanned Combat Air Vehicles are tasked to suppress several Surface-to-Air Missile sites[1]. Each team is assigned to a certain task which consists of suppressing a prescribed sequence of SAM sites. Tasks are executed in parallel and each task is organized in stages. In each stage, each team has to suppress a SAM or only wait that the attacks of other teams are fulfilled. There may be execution dependencies among the stages from different tasks. A task plan is a set of tasks plus execution dependencies. We take a task plan as an input, provided by a high-level decision maker. Dynamic allocation is targeted at load balancing the performance of different teams in the presence of adversarial actions. Load balancing procedure concerns vehicles and some of their attributes, such as fuel, payloads, weapons, etc.

[1]More information about SEAD missions can be found in [64]

Our control framework supports mixed-initiative interactions. The operator can change or update a set of system's parameters based on information that the system is not able to interpret. In our framework the operator is allowed to: 1) update system's parameters based on emergent information; 2) increase or decreased the intensity of the reallocation activity; 3) prevent or enforce some vehicles to move among teams; and 4) adapt the complexity of the problem to be solved to the available computational power and deadlines

Our dynamic allocation/reallocation control problem is decomposed into two problems: a allocation planning problem that gives the minimum number of vehicles needed at each stage in order to perform the remaining stages with a given probability of success, and an execution control problem that balances the performance of teams by reallocating vehicles among them.

## 5.2 SEAD case study

Consider the problem depicted in figure 5.1 where a set of vehicles (UCAVs) $V = \{v_1, v_2, ..., v_{N_v}\}$ has to strike a set of SAMs $S = \{s_1, s_2, ..., s_{N_s}\}$ with a probability $P_f$. Each vehicle $v_i$ is characterized as a tuple:

$$v_i = (VID, fuel, pos, p_{kill}) \tag{5.2.1}$$

where $VID \in \mathbb{N}$ is the vehicle unique identifier, $fuel \in \mathbb{R}$ is the fuel reserves, $pos \in \mathbb{R}^2$ is the 2D position of the vehicle, and $p_{kill} \in [0, 1]$ is the probability of kill, i.e. the probability that the UCAV successfully strikes one SAM [2]. In this work we assume that this probability does not depend on the SAM type. We also assume that a UCAV designated to perform an attack uses all its means to suppress the SAM, i.e. the UCAV fails the attack if and only if it is shot down or runs out of weapons or fuel. We will see further that when a UCAV fails an attack, and the corresponding team is not empty, then another UCAV is designated to perform the attack and so on and so forth until the SAM is suppressed or the team is empty. The SAMs are also characterized as tuples:

$$s_i = (SID, loc) \tag{5.2.2}$$

[2] In our case study we just consider the $VID$, $fuel$, $pos$ and $p_k$ to characterize each vehicle but the formalism is easily extended to other characteristics such as the number of Air-to-Ground Missiles (AGM) available, payload capabilities, among others.

where $SID \in \mathbb{N}$ is the unique identifier of the SAM, and $loc \in \mathbb{R}^2$ is the 2D location of the SAM.

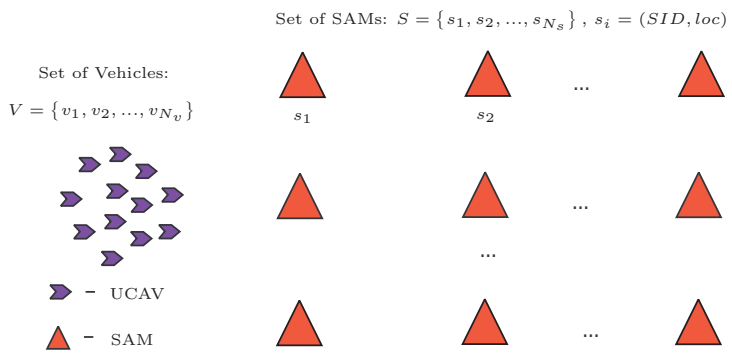



Figure 5.1: The problem: a set of SAMs to be suppressed by a set of vehicles.

The set of SAMs $S$ will be partitioned into several tasks and each task will be assigned to a team of UAVs. All task are divided into $N \in \mathbb{N}$ stages ($k \in K$, $K = \{1, 2, ..., N\}$). Tasks run concurrently and there are precedence constraints among them. Next we will describe the approach to plan a SEAD mission.

## 5.3 Planning SEAD missions

### 5.3.1 Task plan procedure

The task planing procedure is run off-line with the following inputs and output:
**task_plan_procedure**

- Inputs: $S$, $V$, and $world$;
- Output: $tasks$

where $world$ represents the world characteristics such as terrain shape, civil facilities or borders constraints, and $tasks = \{task_1, task_2, ..., task_{N_t}\}$ is the set of tasks. Each task $task_i \in tasks$ is a tuple formed by a sequence of SAMs $(s_1, s_2, ..., s_K)$ and the probability of success ($P_f$) desired to accomplish the task.

$$tasks = \{task_1, task_2, ..., task_{N_t}\}, task_i = ((s_1, ..., s_K), P_f) \tag{5.3.1}$$

where $s_i \in S \cup \emptyset$ ($\emptyset$ is the symbol that represents an empty set), and $s_i \preceq s_{i+1}$[3]. When $s_i = \emptyset$ then there is no SAM to suppress at stage $i$.

In order to define $P_f$ we need to clarify what we mean by successfully accomplish a task. Let us introduce a definition of task's success:

**Definition 5.3.1** (Task success definition)**.** A task $task_i$ is successfully accomplished iff at least one UCAV in the team survives to the overall sequence of attacks.

Thus, according to definition 5.3.1, $P_f$ is the probability that at least one UCAV survives the attack to the last SAM in the sequence.

Figure 5.2 gives a generic example of a task plan.

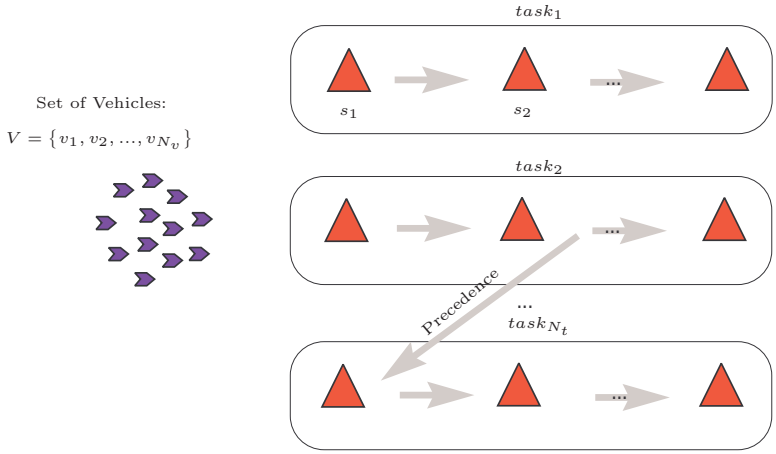



Figure 5.2: Generic example of a task plan.

The execution in a given task may have to wait for another stage to be completed, i.e. some teams have to wait for others before moving to the next target. Such precedence among tasks are modeled by adding $\emptyset$ to the tasks. Let us examine some examples.

**Example 5.1.** *Consider that three teams are tasked to suppress eleven SAMs. Team* 1 *has to suppress* 4 *SAMs such that* $s_1 \preceq s_2 \preceq s_3 \preceq s_4$*. Team* 2 *has to suppress* 4 *SAMs such that* $s_5 \preceq s_6 \preceq s_7 \preceq s_8$*. Finally Team* 3 *has to suppress* 3 *SAMs such that* $s_9 \preceq s_{10} \preceq s_{11}$*. Suppose that there are no precedences constraints among tasks. The SAMs sequences, for each task, are presented in table 5.1.*

[3]The relation $\preceq$ is the precedence relation and is defined as $\{s_i \preceq s_j : s_i, s_j \in S, s_i \text{ precedes } s_j\}$.

| | $k=1$ | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $sams\_sequence_1$ | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $\emptyset$ |
| $sams\_sequence_2$ | $s_5$ | $s_6$ | $s_7$ | $s_8$ | $\emptyset$ |
| $sams\_sequence_3$ | $s_9$ | $s_{10}$ | $s_{11}$ | $\emptyset$ | $\emptyset$ |

Table 5.1: SAMs sequences without precedence among tasks.

**Example 5.2.** *Now suppose that we have precedence among tasks such that $s_1$ precedes $s_5$ and $s_9$, $s_6$ precedes $s_3$ and $s_{10}$ and finally $s_4$ precedes $s_8$ and $s_{11}$. For this case the sequences are shown in table 5.2*

| | $k=1$ | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $sams\_sequence_1$ | $s_1$ | $s_2$ | $\emptyset$ | $s_3$ | $s_4$ | $\emptyset$ | $\emptyset$ |
| $sams\_sequence_2$ | $\emptyset$ | $s_5$ | $s_6$ | $s_7$ | $\emptyset$ | $s_8$ | $\emptyset$ |
| $sams\_sequence_3$ | $\emptyset$ | $s_9$ | $\emptyset$ | $s_{10}$ | $\emptyset$ | $s_{11}$ | $\emptyset$ |

Table 5.2: SAMs sequences with precedence among tasks.

We assume that some high level decision maker invokes the task planning procedure to generate the inputs to the allocation planning procedure that will be discussed in section 5.3.4.

### 5.3.2 Assignment of teams to tasks and vehicles to teams

Each task $task_i$ is assigned to a team of vehicles $team_i \in T$ ($T = \{team_1, team_2, ..., team_{N_t}\}$ such that $team_i \subseteq V$ and $\bigcap_i team_i = \emptyset$). Figure 5.3 shows the assignment of teams to tasks. The initial allocation of vehicles to teams is performed by the allocation procedure

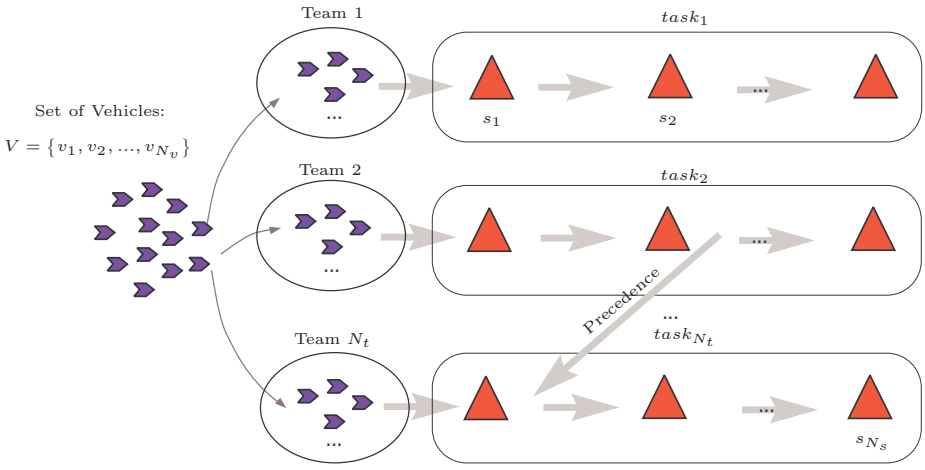



Figure 5.3: Assignment of teams to tasks.

which is addressed in detail in section 5.3.4.

### 5.3.3 Attack model

Before addressing the allocation problem, we model the attack of a team of UCAVs to a sequence of SAMs based on [54]. The attack is performed in several phases. During the first phase the team of UCAVs travels to a safe place close to the SAM. After that, one UCAV is selected to perform the attack maneuver. If the attack succeeds then the team moves to the closest safe place near the next SAM. If the attack is not successful then another UCAV is selected as the attacker. In figure 5.4 we present the FSM of the attack controller. Let us describe each state and transition of the attack controller. The attack

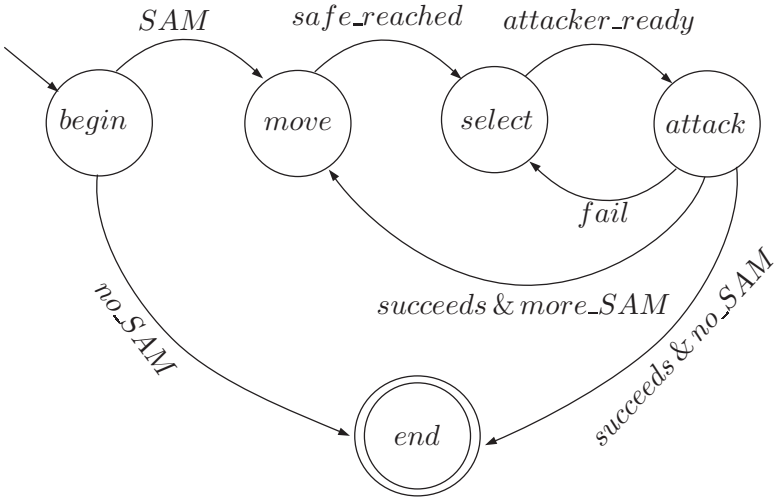



Figure 5.4: FSM of the attack controller.

controller starts at state *begin*; If the team's task has a SAM to strike then the controller moves to state *move* and the team flies towards the SAM, otherwise the controller goes to the marked state *end*; When a team reaches a pre-determined safe place near the SAM, the controller changes to state *select*, where one UCAV is designated to perform the first attempt to strike the SAM; If the attack fails[4] then the controller returns to state *select*; If the attack succeeds and there are more SAMs to strike then the controller moves to state *move*, otherwise it moves to state *end*.

Let us define the set of vehicles that failed the attack as: $V_f \subseteq V$. The set of operational

[4]We assume that a UCAV designated to perform the attack will try to do its best to strike the SAM. Thus, failing an attack means that the UCAV turns to be inoperative in some way: shot down by the SAM, runs out of assets such as missiles and fuel.

vehicles is $\{V - V_f\}$. The assignment of vehicles to perform attacks is done by the function $a : \mathbb{N} \rightarrow \{V - V_f\}$ such that:

$$a(k) = \{v_i \in V : v_i \preceq v_j, \forall v_i, v_j \in \{V - V_f\}, v_i \neq v_j\}. \tag{5.3.2}$$

The function $a$ selects the vehicle to perform the next attack from the set of operational vehicles and according to a pre-defined precedence among vehicles ($a \preceq b$ means that $a$ precedes $b$). We assume that the precedence among vehicles is defined according to their ID, i.e. a vehicle with a lower ID is always selected before a vehicle with a higher ID. This can be mathematically stated as: $v_i \preceq v_j \, iif \, VID(v_i) < VID(v_j) \, \forall i, j \in \mathbb{N}, i \neq j$, where $VID(v_i)$ is the ID of vehicle $v_i$.

We want to characterize the probability of successfully striking a SAM site using the attack procedure. First, in order to calculate such probability, we need a definition of team's success.

**Definition 5.3.2** (Weak success definition)**.** A team of $n$ UCAVs weakly succeeds in attacking one SAM iif at least one of the $n$ UCAVs survives the attack.

We define $u_k$ as the number of vehicles at stage $k$. Thus, according to definition 5.3.2, the team succeeds iif $u_{k+1} \geq 1$ (note that if the team is on stage $k$, the definition 5.3.2 claims that the team succeeds if after the attack (stage $k+1$) there is at least one vehicle).

Assuming that the attacks are independent, that all UCAVs have the same probability of successfully destroying a SAM site ($p_{kill}$), and that the UCAVs are selected to perform the attack according to function $a$, we can define the probability of $n$ UCAVs strikes successfully one SAM as:

$$\begin{aligned} & \text{Prob}(u_{k+1} \geq 1 | u_k = n) = p_{kill} + p_{kill}(1 - p_{kill}) + p_{kill}(1 - p_{kill})^2 + ... + p_{kill}(1 - p_{kill})^{n-1} \\ \Leftrightarrow \quad & \text{Prob}(u_{k+1} \geq 1 | u_k = n) = \sum_{i=0}^{n-1} p_{kill}(1 - p_{kill})^i \end{aligned} \tag{5.3.3}$$

In other words, equation 5.3.3 states that the probability of interest is given by the probability that the first attacker succeeds, or the first one fails and the second succeeds, or the second also fails and the third succeeds, and so on and so forth until we reach the last possibility where $n - 1$ UCAVs fail and the last one succeeds. Remind that failing doesn't always imply that a UCAV was shot by a SAM but that a UCAV turned to be inoperative in some way (shot down by the SAM or runs out of missiles or fuel).

We designated the definition 5.3.2 as "weak success definition" because it only guarantees that one UCAV survives the attack. Let us give a stronger success definition:

**Definition 5.3.3** (m-Strong success definition)**.** A team of $n$ UCAVs strongly succeeds in attacking one SAM iif at least $m \leq n$ UCAVs survive the attack.

Definition 5.3.3 imposes that at least $m$ UCAVs survives the attack. Such definition could also be viewed as a generalization of the definition 5.3.2. Let us now redefine equation 5.3.3 according to definition 5.3.3:

$$\text{Prob}(u_{k+1} \geq m | u_k = n) = \sum_{i=0}^{n-m} p_{kill}(1 - p_{kill})^i \tag{5.3.4}$$

Equation 5.3.4 considers only the attack of a team of $n$ UCAVs to one SAM site. Let us now give the general case where several SAM sites are attacked in sequence. Consider that a task has $M$ SAMs and the team assigned to that task is on stage $k$. Thus, the team has to strike $M + 1 - k$ SAMs and at least $n_f$ UCAVs should survive the attacks (i.e. at least $n_f$ UCAVs should reach the stage $M + 1$). Using the same reasoning as in equation 5.3.3 we have:

$$\begin{aligned}
&\text{Prob}(u_{M+1} \geq 1 | u_k = n) = p_{kill}^{M+1-k} + p_{kill}^{M+1-k}(1 - p_{kill})\text{C}_1^{M+1-k} + \\
&+p_{kill}^{M+1-k}(1 - p_{kill})^2 \cdot (\text{C}_1^{M+1-k} + \text{C}_2^{M+1-k}) + \dots \\
&\dots + p_{kill}^{M+1-k}(1 - p_{kill})^{n-n_f} \sum_{j=1}^{n-n_f} \text{C}_j^{M+1-k} \\
\Leftrightarrow \quad &\text{Prob}(u_{M+1} \geq 1 | u_k = n) = \sum_{i=0}^{n-n_f} p_{kill}^{M+1-k}(1 - p_{kill})^i \sum_{j=1}^{i} \text{C}_j^{M+1-k}
\end{aligned} \tag{5.3.5}$$

where $\text{C}_k^n = \frac{n!}{k!(n-k)!}$ is the number of possible combinations that, in a set of $n$ elements, we can choose $k$ elements. Note that the terms multiplied by $p_{kill}$ in equation 5.3.3 are now multiplied by $p_{kill}^{M+1-k}$ in order to include the attacks to $M + 1 - k$ SAMs. The term $\sum_{j=1}^{i} \text{C}_j^{M+1-k}$ gives the different combinations of SAMs sites where $i$ UCAVs failed, given that those UCAVs attacked $M + 1 - k$ SAM sites. For example, if 3 UCAVs failed during the attacks of 3 SAM sites, then the possible combinations are: all the 3 UCAVs failed attacking the same SAM ($\text{C}_1^3$); two UCAVs failed attacking the same SAM and the other UCAV failed attacking a different SAM ($\text{C}_2^3$); and all UCAVs failed attacking different SAM sites ($\text{C}_3^3$). Note that, if the sequence of UCAVs selected to do the attacks weren't pre-definite then the possible combinations would be higher. Equation 5.3.5 will be useful later to define the allocation plan.

### 5.3.4 The allocation planning procedure

The allocation planning procedure is inspired in the Model Predictive Control framework applied to military operations presented in [65]. In this work, prediction models are updated to describe the dynamics of degradation of assets over time as a result of combat activities.

In our reallocation problem, the allocation procedure provides the initial allocation of vehicles to teams that ensures that the last SAMs in each task are eliminated with probability greater than $P_f$. This procedure also provides the minimum number of vehicles needed at each stage, i.e. for each stage $k$ how many vehicles each team needs to perform the remaining $N-k$ stages in order to guarantee a final probability of success greater then $P_f$. This information will be useful later to evaluate the teams' performance. Figure 5.5 depicts a generic example of the allocation of vehicles to teams for each stage.

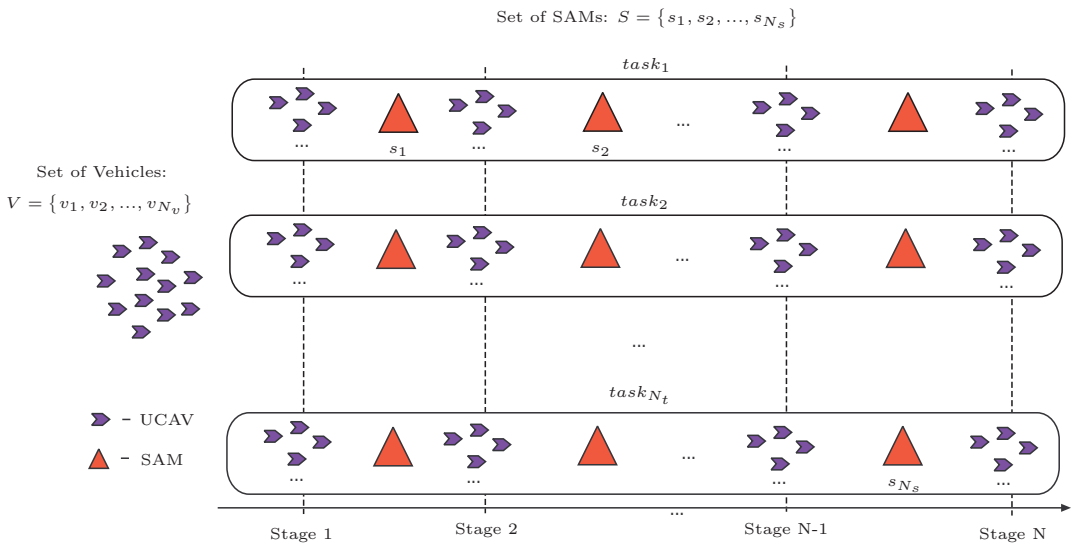



Figure 5.5: Multi-stage resource allocation.

The output of the allocation planning procedure is presented in the form of the matrix $\mathbf{P}$ where each element $P : |K| \times |T| \rightarrow \mathbb{N}_0$ corresponds to the minimum number of vehicles needed for team $t$ performs the remaining $N-k$ stages so that the last SAM is destroyed with a probability of at least $P_f$. In summary, the inputs and outputs of the allocation planning procedure are:

**allocation_procedure**

- Inputs: $tasks$, $p_{kill}$, $P_f$;

- Output: **P**.

where *tasks* is the set of tasks and $p_{kill}$ is the probability of one UCAV to strike successfully one SAM[5].

Using the equation 5.3.4 we can find the minimum number of vehicles needed at stage $k$ ($u_k^*$) in order to perform the remaining task with at least the desired probability of success $P_f$. Suppose that a team has to strike $M$ SAMs. Let us define a cost function for each stage $k \in \{1, ..., M+1\}$ using equation 5.3.5:

$$J_k(u_k) = \text{Prob}(u_{M+1} \geq u_{M+1}^* | u_k) \tag{5.3.6}$$

where $u_k$ is the number of vehicles at stage $k$. Now we have to find all $u_k^*$ such that $J_k^* \geq P_f$ for all $k$ ($J_k^* = J_k(u_k^*)$).

We model the allocation problem as the following optimization problem:

$$u_k^* = \arg\min_u J_k(u) \tag{5.3.7}$$

$$\text{subject to} \quad J_k(u) \geq P_f \tag{5.3.8}$$

$$u_{M+1}^* = 1.$$

If we want to impose a strong definition of task's success such that $n_f$ vehicles reach the end of the stage rather then just one, then we have to set $u_{M+1}^* = n_f$.

In figure 5.6 we present a generic example for this procedure. where $J_k^*$ is the optimum

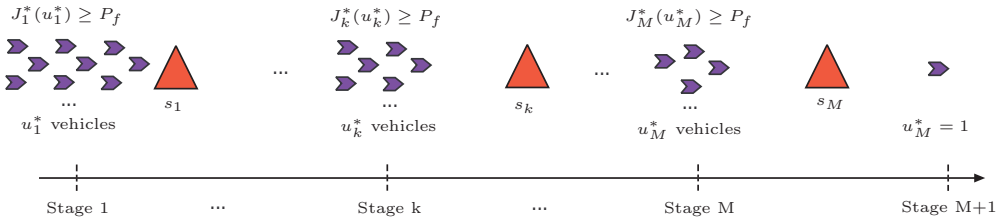



Figure 5.6: Illustration of the probability success evaluated at each stage.

value of the cost function at stage $k$.

**Example 5.3.** *Let us take an example. Suppose that a task composed by 5 SAMs ($M = 5$) has to be accomplished with a probability of 80%. The probability $p_{kill} = 0.7$ is uniform among all UCAVs.*

[5]For the sake of simplicity of the notation we consider that $p_{kill}$ is homogeneous among all UCAVs and $P_f$ is the same for all tasks.

*The table 5.3 shows the optimum number of vehicles per stage ($u_k^*$) and the optimum cost function $J_k^* = J_k(u_k^*)$ for each SAM $k$.*

| $k$: | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $u_k^*$ | **5** | **4** | **3** | **3** | **2** | **1** |
| $J_k^*$ | $0.8014 > 0.8$ | $0.8351 > 0.8$ | $0.8369 > 0.8$ | $0.9163 > 0.8$ | $0.9100 > 0.8$ | – |

Table 5.3: Plan for a team to suppress 5 SAMs with $p_{kill} = 0.7$, $n_f = 1$ and $P_f = 0.8$.

**Example 5.4.** *Consider the problem presented in example 5.1 where 3 teams have to suppress eleven SAMs with $P_f = 0.8$ and $p_{kill} = 0.7$. The matrix $\mathbf{P}$ in this case will be:*

$$\mathbf{P} = \begin{array}{c} \\ k=1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{bmatrix} 4 & 4 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 2 \\ 2 & 2 & 1 \\ 1 & 1 & 0 \end{bmatrix} \end{array} \tag{5.3.9}$$

**Example 5.5.** *Let us present the plan for example 5.2 where several precedence constraints are considered among attacks (see table 5.2).*

$$\mathbf{P} = \begin{array}{c} \\ k=1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccc} task_1 & task_2 & task_3 \end{array} \\ \begin{bmatrix} 4 & 4 & 3 \\ 3 & 4 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \\ 2 & 2 & 2 \\ 1 & 2 & 2 \\ 0 & 1 & 1 \end{bmatrix} \end{array} \tag{5.3.10}$$

*When a team is holding at a certain stage due to a precedence (symbolized with an $\emptyset$), it doesn't change the number of UCAVs needed to perform the remaining task. When a $\emptyset$ is inserted in a stage, then the plan on that stage remains the same of the next stage. Thus, the initial allocation of vehicles to teams is not affected by the precedences constraints (note that the first row of matrixes 5.3.9 and 5.3.10 are equal).*

The operator is able to tune the probability $P_f$ establishing a trade-off between the effectiveness of mission and the number of assets needed. We will see later that our framework allows the operator to ask for a replan, to change some parameters based on his experience, or on some information that wasn't included into the planning procedure. These mixed-initiative interactions will be explored at section 5.4.8.

## 5.4 Execution control problem with mixed initiative interactions

In this section we model the reallocation problem.

### 5.4.1 Vehicles reallocation problem

The execution will be conducted in an adversarial environment where UCAVs may be brought down as a consequence of the actions of the adversary. Thus, it is possible that execution may not go as planned. Some teams may lack UCAVs to execute their tasks, while others may have them in excess. This could happen due to the uncertainty of the attacks and also due to the fact that UCAVs could be destroyed by some unknown threat ("pop-up" threats). Thus, the overall performance may be improved by transferring UCAVs among teams. Two important issues arise regarding the reallocation problem: some reallocations may not be feasible (e.g. a vehicle may not have enough fuel to transit to another team); and the reallocation of vehicles may have a transition cost. The execution control problem consists of balancing the performance of teams by exchanging vehicles among them while the transition costs are minimized under additional constraints that we will explain below.

Figure 5.7 depicts our control problem.

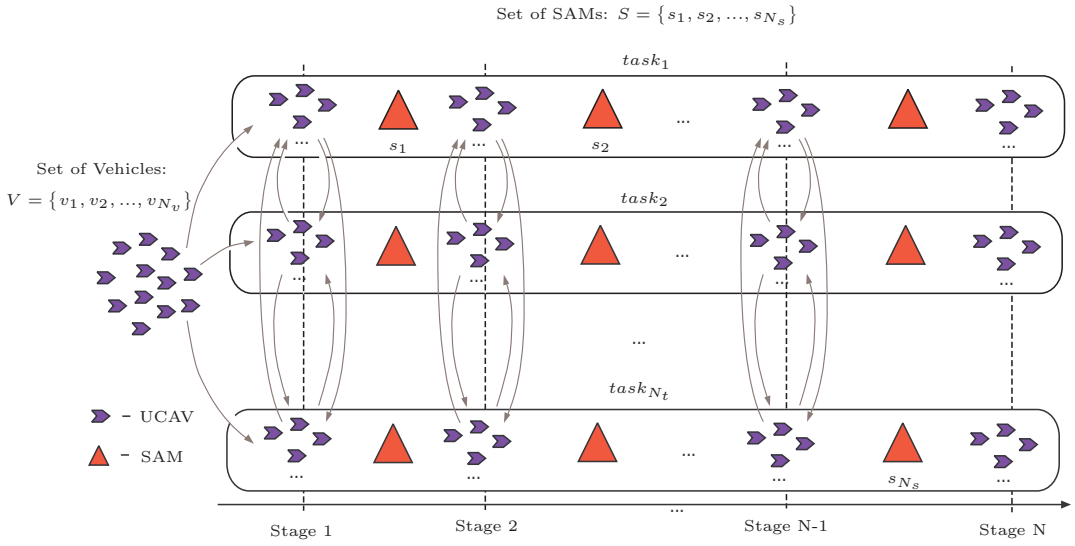



Figure 5.7: Multi-stage resource allocation.

### 5.4.2 System dynamics

Let $V$ (set of vehicles) and P (plan) be given. We define the allocation of vehicles to teams at stage $k$ as a matrix $\mathbf{Z}_k$ such that each element $Z_k : \{1 \dots |T|\} \times \{1 \dots |V|\} \rightarrow \{0, 1\}$ where:

$$Z_k(i, j) = \begin{cases} 1, & \text{iff vehicle } j \text{ is in team } i; \\ 0, & \text{otherwise.} \end{cases} \tag{5.4.1}$$

We define the transition-vehicle matrix $\boldsymbol{\mathcal{A}}_k$ with elements $\mathcal{A}_k : \{1 \dots |T_r|\} \times \{1 \dots |V|\} \rightarrow \{0, 1\}$ where $T_r = \{t_r = (t_i, t_j) : \forall i, j \in T, i \neq j\}$ is the set of all possible transitions among teams and:

$$\mathcal{A}_k(i, j) = \begin{cases} 1, & \text{iff vehicle } j \text{ is reallocated using transition } i; \\ 0, & \text{otherwise.} \end{cases} \tag{5.4.2}$$

Entries modeling forbidden transitions are set to zero. The non-zero terms are the decision variables for any problem regarding the exchange of vehicles. To guarantee that each vehicle is allocated to, at most, one transition we introduce the following conservation constraint:

$$\boldsymbol{\mathcal{A}}'_k \cdot \mathbf{1} \leq \mathbf{1} \tag{5.4.3}$$

where $\mathbf{1}$ denotes the vector of ones. Basically, constraint 5.4.3 ensures that the summation of the elements of each column of $\boldsymbol{\mathcal{A}}_k$ is zero (the vehicle of the corresponding column is not reallocated) or one (the vehicle is just reallocated through one transition).

Consider the matrices $\boldsymbol{\mathcal{N}}_{sk} \in \{0, 1\}^{T \times T_r}$ and $\boldsymbol{\mathcal{N}}_{dk} \in \{0, 1\}^{T \times T_r}$ such that each element is:

$$\mathcal{N}_{sk}(i, j) = \begin{cases} 1, & \text{if team } i \text{ is the source team of the transition } j; \\ 0, & \text{otherwise.} \end{cases} \tag{5.4.4}$$

$$\mathcal{N}_{dk}(i, j) = \begin{cases} 1, & \text{if team } i \text{ is the destination team of the transition } j; \\ 0, & \text{otherwise.} \end{cases} \tag{5.4.5}$$

Note that the columns of $\boldsymbol{\mathcal{N}}_{sk}$ and $\boldsymbol{\mathcal{N}}_{dk}$ represent the transitions among teams and, for each column $j$, the rows with elements set to 1 are, respectively, the source and destination teams for transition $j$. Consider now the matrix $\boldsymbol{\mathcal{N}}_k \in \{0, 1\}^{T \times T_r}$ such that:

$$\boldsymbol{\mathcal{N}}_k = \boldsymbol{\mathcal{N}}_{sk} - \boldsymbol{\mathcal{N}}_{dk}. \tag{5.4.6}$$

The elements of $\boldsymbol{\mathcal{N}}_k$ label the source teams as 1 and destinations teams as $-1$ for each transition. The matrix $\boldsymbol{\mathcal{N}}_k$ is usually known in the network flow literature as the node-arc incidence matrix [66].

Using the transition-vehicle matrix $\boldsymbol{\mathcal{A}}_k$ and the node-arc incidence matrix $\boldsymbol{\mathcal{N}}_k$ we can state the allocation matrix after a reallocation action as:

$$\mathbf{Z}_k^r = \mathbf{Z}_k - \boldsymbol{\mathcal{N}}_k \cdot \boldsymbol{\mathcal{A}}_k \tag{5.4.7}$$

Let us take a reallocation example.

**Example 5.6.** *Consider the initial vehicle allocation:*

$$\mathbf{Z}_k = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \end{array} \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array} \tag{5.4.8}$$

*The transition-vehicle matrix will be:*

$$\boldsymbol{\mathcal{A}}_k = \begin{array}{c} \\ (t_1,t_2) \\ (t_1,t_3) \\ (t_2,t_1) \\ (t_2,t_3) \\ (t_3,t_1) \\ (t_3,t_2) \end{array} \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{bmatrix} a_{1,1} & 0 & 0 & 0 & 0 & 0 \\ a_{2,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{3,2} & a_{3,3} & 0 & 0 & 0 \\ 0 & a_{4,2} & a_{4,3} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{5,4} & a_{5,5} & a_{5,6} \\ 0 & 0 & 0 & a_{6,4} & a_{6,5} & a_{6,6} \end{bmatrix} \end{array} \tag{5.4.9}$$

*The matrices 5.4.4, 5.4.5 and 5.4.6 will be:*

$$\boldsymbol{\mathcal{N}}_{sk} = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \end{array} \begin{array}{c} \begin{array}{cccccc} (t_1,t_2) & (t_1,t_3) & (t_2,t_1) & (t_2,t_3) & (t_3,t_1) & (t_3,t_2) \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \tag{5.4.10}$$

,

$$\boldsymbol{\mathcal{N}}_{dk} = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \end{array} \begin{array}{c} \begin{array}{cccccc} (t_1,t_2) & (t_1,t_3) & (t_2,t_1) & (t_2,t_3) & (t_3,t_1) & (t_3,t_2) \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array} \tag{5.4.11}$$

*and*

$$\boldsymbol{\mathcal{N}}_k = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \end{array} \begin{array}{c} \begin{array}{cccccc} (t_1,t_2) & (t_1,t_3) & (t_2,t_1) & (t_2,t_3) & (t_3,t_1) & (t_3,t_2) \end{array} \\ \begin{bmatrix} 1 & 1 & -1 & 0 & -1 & 0 \\ -1 & 0 & 1 & 1 & 0 & -1 \\ 0 & -1 & 0 & -1 & 1 & 1 \end{bmatrix} \end{array} \tag{5.4.12}$$

*We are now able to calculate $\boldsymbol{Z}_k^r$ from 5.4.7:*

$$\mathbf{Z}_k^r = \begin{bmatrix} 1-a_{1,1}-a_{2,1} & a_{3,2} & a_{3,3} & a_{5,4} & a_{5,5} & a_{5,6} \\ a_{1,1} & 1-a_{3,2}-a_{4,2} & 1-a_{3,3}-a_{4,3} & a_{6,4} & a_{6,5} & a_{6,6} \\ a_{2,1} & a_{4,2} & a_{4,3} & 1-a_{5,4}-a_{6,4} & 1-a_{5,5}-a_{6,5} & 1-a_{5,6}-a_{6,6} \end{bmatrix} \tag{5.4.13}$$

The adversarial behavior, where vehicles may be brought down, is modeled by subtracting vehicles from the allocation matrix. The resulting allocation matrix at stage $k+1$ is:

$$\mathbf{Z}_{k+1} = \mathbf{Z}_k^r - \mathbf{W}_k \tag{5.4.14}$$

where $\mathbf{W}_k$ is the uncertainty matrix with elements $W_k : T \times V \rightarrow \{0, 1\}$ and

$$W_k(i,j) = \begin{cases} 1, & \text{iff vehicle } j \text{ of team } i \text{ was shot down;} \\ 0, & \text{otherwise.} \end{cases} \tag{5.4.15}$$

Each element of matrix $\boldsymbol{W}_k$ is a stochastic variable such that, if vehicle $j$ in team $i$ is performing an attack to a SAM then $W_k(i,j) = 1$ with the probability:

$$\text{Prob}(W_k(i,j) = 1) = 1 - p_{kill}. \tag{5.4.16}$$

If the vehicle $j$ is not performing an attack then $W_k(i,j) = 0$. If the vehicle $j$ is performing a reallocation from team $i$ to team $l$ then:

$$\text{Prob}(W_k(i,j) = 1) = 1 - p_s(i,j,l), \tag{5.4.17}$$

where $p_s(i,j,l)$ is the probability of success of vehicle $j$ transit from team $i$ to team $l$. We assume that equation 5.4.17 arises from pop-up threats. Other sources of uncertainty are easily modeled in this framework either by changing $p_s$ or by subtracting other matrices (similar to $W_k$) in equation 5.4.14.

The reallocation of vehicles among teams may be constrained in several ways. This is discussed next.

## 5.4.3 Constraints

### 5.4.3.1 Vehicles' constraints

The reallocations may be constrained by vehicles' attributes (eg. type of vehicle) and their state (eg. available fuel). Consider the following model for fuel constraints at stage $k$:

$$\boldsymbol{\mathcal{A}}'_k \cdot \boldsymbol{c}_{fk} \leq \boldsymbol{f}_{rk} \tag{5.4.18}$$

where $\boldsymbol{c}_{fk} \in \mathbb{R}^{|T_r|}$ is a vector with fuel cost of transitions and $\boldsymbol{f}_{rk} \in \mathbb{R}^{|V|}$ is a vector with the fuel reserves of vehicles. Each element of the fuel cost vector is calculated as follows:

$$c_{fk}(t_r = (t_i, t_j)) = f_c \cdot \left( \|f_{av}(t_i) - f_{av}(t_j)\|_2 + \left\|f_{av}(t_j) - loc_{M_j}\right\|_2 \right) \tag{5.4.19}$$

where $f_{av} : T \to \mathbb{R}^2$ maps teams to their average 2D position, $loc_{M_j}$ is the 2D position of the last SAM in the task of team $t_j$, $f_c$ is the rate of fuel consumption per unit of distance and $\|\cdot\|_2$ is the Euclidean norm. Equation 5.4.18 constraints the reallocation a vehicle to have fuel enough to perform the transition within teams and, in addition, to reach the end of the task of the destination team.

**Example 5.7.** *Let us take the example 5.6 now with the following fuel constraints:*

$$\boldsymbol{c}_{fk} = \begin{matrix} (t_1,t_2) \\ (t_1,t_3) \\ (t_2,t_1) \\ (t_2,t_3) \\ (t_3,t_1) \\ (t_3,t_2) \end{matrix} \begin{bmatrix} 0.4 \\ 0.5 \\ 0.4 \\ 0.1 \\ 0.5 \\ 0.1 \end{bmatrix} \tag{5.4.20}$$

*and the follow fuel reserves of vehicles:*

$$\boldsymbol{f}_{rk} = \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix} \begin{bmatrix} 0.3 \\ 0.6 \\ 0.6 \\ 0.5 \\ 0.5 \\ 0.4 \end{bmatrix} \tag{5.4.21}$$

$$\boldsymbol{\mathcal{A}}_k^T \cdot \boldsymbol{c}_{fk} \leq \boldsymbol{f}_{rk}$$

$$\begin{bmatrix} a_{1,1} & 0 & 0 & 0 & 0 & 0 \\ a_{2,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{3,2} & a_{3,3} & 0 & 0 & 0 \\ 0 & a_{4,2} & a_{4,3} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{5,4} & a_{5,5} & a_{5,6} \\ 0 & 0 & 0 & a_{6,4} & a_{6,5} & a_{6,6} \end{bmatrix}^T \cdot \begin{bmatrix} 0.4 \\ 0.5 \\ 0.4 \\ 0.1 \\ 0.5 \\ 0.1 \end{bmatrix} \leq \begin{bmatrix} 0.3 \\ 0.6 \\ 0.6 \\ 0.5 \\ 0.5 \\ 0.4 \end{bmatrix}$$

$$\begin{bmatrix} 0.4\,a_{1,1} + 0.5\,a_{2,1} \\ 0.4\,a_{3,2} + 0.1\,a_{4,2} \\ 0.4\,a_{3,3} + 0.1\,a_{4,3} \\ 0.5\,a_{5,4} + 0.1\,a_{6,4} \\ 0.5\,a_{5,5} + 0.1\,a_{6,5} \\ 0.5\,a_{5,6} + 0.1\,a_{6,6} \end{bmatrix} \leq \begin{bmatrix} 0.3 \\ 0.6 \\ 0.6 \\ 0.5 \\ 0.5 \\ 0.4 \end{bmatrix} \tag{5.4.22}$$

*We can see that transitions corresponding to $a_{1,1}$, $a_{2,1}$ and $a_{5,6}$ are no more feasible. Thus, the constrained matrix $\mathcal{A}_k$ and the respective decision variables are given as follow:*

$$\boldsymbol{\mathcal{A}}_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{3,2} & a_{3,3} & 0 & 0 & 0 \\ 0 & a_{4,2} & a_{4,3} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{5,4} & a_{5,5} & 0 \\ 0 & 0 & 0 & a_{6,4} & a_{6,5} & a_{6,6} \end{bmatrix} \tag{5.4.23}$$

Using the above procedure it is easy to introduce other constraints due to vehicle characteristics. However, the state of the system tends to grow with the number of characteristics and, thus, the complexity to solve a reallocation problem will grow too. This fact puts more emphasis on the necessity of having the operator in the control loop, particularly during the planning procedure where he can decide what vehicles' characteristics should be considered in the problem.

Another way that the operator could deal with the complexity of the problem is by forbidding some reallocations to happen and thus decrease the action space. Next we introduce a constraint that gives the operator this degree of freedom.

#### 5.4.3.2 Operator constraints

Consider the following constraint:

$$\boldsymbol{\mathcal{A}}'_k \cdot \mathbf{1} \leq \boldsymbol{b}_v, \tag{5.4.24}$$

where $\boldsymbol{b}_v \in \{0,1\}^{|V|}$ is a binary vector such that if $b_v(i) = 1$ then vehicle $i$ is able to reallocate, if $b_v(i) = 0$ the vehicle is unable to reallocate. With this constraint the operator can easily prevent some vehicles to be reallocated by simply changing the elements of vector $\boldsymbol{b}_v$. Besides allowing the operator to reduce the complexity of the problem by reducing the action space, this constraint can also be used with a tactical meaning, i.e. the operator may want to ensure that some vehicles, with a certain role in the team (e.g. a leader), will never be reallocated.

### 5.4.4 Performance considerations

We evaluate the teams' performance at each stage by simply comparing the number of vehicles that the teams actually have with the number of vehicles that we expected that they should have at that stage. This is measured, at each stage and for all teams, with the performance vector $\boldsymbol{\gamma}_k : \mathbb{N}^{|T|} \times \mathbb{N}^{|T|} \rightarrow \mathbb{Z}^{|T|}$ where:

$$\boldsymbol{\gamma}_k = \mathbf{Z}_k \cdot \mathbf{1} - \boldsymbol{p}'_k. \tag{5.4.25}$$

The components of $\boldsymbol{\gamma}_k$ are the performance of teams at stage $k$, $\mathbf{Z}_k \cdot \mathbf{1}$ is a vector with the number of vehicles of each team at stage $k$ and $\boldsymbol{p}_k$ is the $k^{th}$ row of the matrix $\mathbf{P}$ and stands for the number of vehicles initially planned for each team at stage $k$.

The vector 5.4.25 can be rearranged as the sum of the two following vectors:

$$\boldsymbol{\gamma}_k = \boldsymbol{\gamma}_k^s + \boldsymbol{\gamma}_k^d \tag{5.4.26}$$

$$\gamma_k^s(i) = \begin{cases} \gamma_k(i), & \text{if } \gamma_k(i) > 0,\ \forall i \in T; \\ 0, & \text{otherwise.} \end{cases} \tag{5.4.27}$$

$$\gamma_k^d(i) = \begin{cases} \gamma_k(i), & \text{if } \gamma_k(i) < 0,\ \forall i \in T; \\ 0, & \text{otherwise.} \end{cases} \tag{5.4.28}$$

where $\boldsymbol{\gamma}_k^s$ and $\boldsymbol{\gamma}_k^d$ give, respectively, the number of vehicles that the suppliers can deliver and the number of vehicles that the demanders want to receive.

According to their performance, we label teams as:

1. Suppliers - $T_s = \{t : \forall t \in T,\ \gamma_k(t) > 0\}$;
2. Demanders - $T_d = \{t : \forall t \in T,\ \gamma_k(t) < 0\}$;
3. Neutrals - $T_n = \{t \forall t \in T,\ \gamma_k(t) = 0\}$.

Regarding the above partition it is obvious that $T = T_s \cup T_d \cup T_n$ and $T_s \cap T_d \cap T_n = \emptyset$.

#### 5.4.4.1 Suppliers and demanders

One way to decrease size of the action space, and thus the complexity of the problem, is to allow reallocations only among suppliers teams ($T_s$) and demanders teams ($T_d$). This is done with the following two constraints:

$$\boldsymbol{\mathcal{N}}_{sk} \cdot \boldsymbol{\mathcal{A}}_k \cdot \mathbf{1} \leq \boldsymbol{\gamma}_k^s + \boldsymbol{b}_k^s; \tag{5.4.29}$$

$$\boldsymbol{\mathcal{N}}_{dk} \cdot \boldsymbol{\mathcal{A}}_k \cdot \mathbf{1} \leq -\boldsymbol{\gamma}_k^d + \boldsymbol{b}_k^d. \tag{5.4.30}$$

where $\boldsymbol{b}_k^s \in \mathbb{Z}^{|T|}$ is the supply boundary and $\boldsymbol{b}_k^d \in \mathbb{Z}^{|T|}$ is the demand boundary. These boundaries give the operator the ability to relax or stiffen the constraints. If the element $b_k^s(t \in T_s) > 0$ then the team $t$ is allowed to deliver more vehicles than what it was supposed to (if $b_k^s(t \in T_s) < 0$ then we are constraining even more the number of vehicles to deliver). With $b_k^d$ we can do the same thing with the number of vehicles that a demander team can receive. With the two vectors we can actually turn a neutral team into a demander or into a supplier. In conclusion, they give the possibility to the operator to increase or decrease the flexibility of the balancing procedure.

**Example 5.8.** *Let us take the example 5.6. The transition-vehicle matrix is given by equation 5.4.9. Let us assume that the number of vehicles planned for teams are:*

$$\boldsymbol{p}'_k = \begin{matrix} t_1 \\ t_2 \\ t_3 \end{matrix} \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}. \tag{5.4.31}$$

*Looking to matrix 5.4.8 we realize that:*

$$\mathbf{Z}_k \cdot \mathbf{1} = \begin{matrix} t_1 \\ t_2 \\ t_3 \end{matrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}. \tag{5.4.32}$$

*Thus,*

$$\boldsymbol{\gamma}_k = \begin{matrix} t_1 \\ t_2 \\ t_3 \end{matrix} \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix} \tag{5.4.33}$$

*From 5.4.29 and 5.4.30 we have:*

$$\boldsymbol{\gamma}^s_k = \begin{matrix} t_1 \\ t_2 \\ t_3 \end{matrix} \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \tag{5.4.34}$$

*and*

$$\boldsymbol{\gamma}^d_k = \begin{matrix} t_1 \\ t_2 \\ t_3 \end{matrix} \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}. \tag{5.4.35}$$

*We are now able to calculate the constraints 5.4.29 and 5.4.30 (we assume here that $\boldsymbol{b}^s_k = \boldsymbol{b}^d_k = \mathbf{0}$ where $\mathbf{0}$ is a vector of zeros):*

$$\begin{gathered}
\boldsymbol{\mathcal{N}}_{sk} \cdot \boldsymbol{\mathcal{A}}_k \cdot \boldsymbol{1} \leq \boldsymbol{\gamma}^s_k \\
\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{1,1} & 0 & 0 & 0 & 0 & 0 \\ a_{2,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{3,2} & a_{3,3} & 0 & 0 & 0 \\ 0 & a_{4,2} & a_{4,3} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{5,4} & a_{5,5} & a_{5,6} \\ 0 & 0 & 0 & a_{6,4} & a_{6,5} & a_{6,6} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \\
\begin{bmatrix} a_{1,1} + a_{2,1} \\ a_{3,2} + a_{4,2} + a_{3,3} + a_{4,3} \\ a_{5,4} + a_{6,4} + a_{5,5} + a_{6,5} + a_{5,6} + a_{6,6} \end{bmatrix} \leq \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}
\end{gathered} \tag{5.4.36}$$

$$\boldsymbol{\mathcal{N}}_{dk} \cdot \boldsymbol{\mathcal{A}}_k \cdot \boldsymbol{1} \leq -\boldsymbol{\gamma}_k^d$$

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} a_{1,1} & 0 & 0 & 0 & 0 & 0 \\ a_{2,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{3,2} & a_{3,3} & 0 & 0 & 0 \\ 0 & a_{4,2} & a_{4,3} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{5,4} & a_{5,5} & a_{5,6} \\ 0 & 0 & 0 & a_{6,4} & a_{6,5} & a_{6,6} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \leq - \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} a_{3,2} + a_{3,3} + a_{5,4} + a_{5,5} + a_{5,6} \\ a_{1,1} + a_{6,4} + a_{6,5} + a_{6,6} \\ a_{2,1} + a_{4,2} + a_{4,3} \end{bmatrix} \leq - \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}. \tag{5.4.37}$$

*Analyzing the constraints 5.4.36 and 5.4.37 we can see that the only non-zero terms of* $\boldsymbol{\mathcal{A}}$ *are:* $a_{3,2}$, $a_{3,3}$, $a_{5,4}$, $a_{5,5}$ *and* $a_{5,6}$. *Thus, the constrained matrix* $\boldsymbol{\mathcal{A}}$ *will be:*

$$\boldsymbol{\mathcal{A}}_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{3,2} & a_{3,3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{5,4} & a_{5,5} & a_{5,6} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.4.38}$$

Let us now take an example on the use of the variables $\boldsymbol{b}^s$ and $\boldsymbol{b}^d$.

**Example 5.9.** *Consider five teams that at a certain stage have the performances measures given by the following* $\gamma$ *vector:*

$$\boldsymbol{\gamma} = \begin{bmatrix} 3 \\ -2 \\ -1 \\ 4 \\ -3 \end{bmatrix} \tag{5.4.39}$$

*Consider that the operator wants to reduce the reallocation effort by just allowing the transitions of vehicles if the absolute value of the performance measure is greater than one. This can be done with the following* $b^s$ *and* $b^d$ *vectors:*

$$\boldsymbol{b}^s = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{5.4.40}$$

$$\boldsymbol{b}^d = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{5.4.41}$$

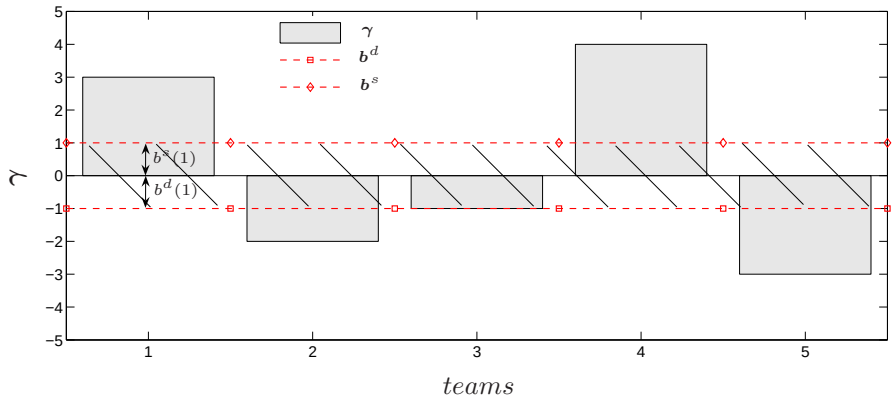



Figure 5.8: Example of the use of boundary layers $\boldsymbol{b}^s$ and $\boldsymbol{b}^d$.

*The figure 5.8 depicts this example. In figure 5.8 we can see that* $\boldsymbol{b}^s$ *and* $\boldsymbol{b}^d$ *form a boundary layer such that if* $\boldsymbol{\gamma}$ *is inside this boundary layer, then no reallocations will happen. In this simple example this boundary takes the same value for all teams. Notice that the boundary can be independent for each team.*

### 5.4.5 Reallocation criteria

We identify three different (but not independent) criteria in the reallocation problem:

1. Difference of performance among teams (balancing objective);
2. Transition costs;
3. Risk of losing vehicles.

The above criteria can have conflicting objectives. Actually, the first criteria will conflict with the second and the third: to balance teams we need to move vehicles among them, this may lead to transition costs and risk of loosing vehicles (a vehicle could be shot by a pop-up threat during a transition).

Our problem can be defined as Multi-Objective Markov Decision Problem (MOMDP). Recently, multi-objective optimization techniques, such as Goal Programming (GP), have

been adapted to solve MOMDPs. [67] presents a GP algorithm with dynamic goals able to take into account the evolution along the process period.

A useful tool to analyze multi-objective optimization problems is the well known Pareto-optimal set. The Pareto-optimal set of a multi-objective optimization problem is the set of solutions such that it is not possible to find a better solution that improves any objective without at the same time worsen another. Several techniques have been presented to solve multi-objective optimization problems and normally they seek Pareto-optimal solutions. The solutions that are not Pareto-optimal are called dominated solutions while the solutions inside the Pareto-optimal set are called not dominated solutions. In the case of MOMDPs the Pareto-Optimal solutions are evaluated taking into account the overall control policy rather than looking for just one control action. A definition of Pareto-Optimal policies is provided in [68].

We use a weighted cost function to solve the multi-criteria behavior of the reallocation control problem. We choose this solution because it is computationally light and because we can give to the operator the task of tuning the weights according to his experience. In fact, with these weights the operator can tune the system in order to better fit his risk expectations. A similar example is a financial assets portfolio where an investor allocates his assets in diversified applications according to his risk profile.

Next we define the weighted cost function for our reallocation problem.

#### 5.4.5.1 Weighted cost function

The cost function $c_k : X_k \times A_k \rightarrow \mathbb{R}^+$ maps the state and the control action at each stage into a positive real number. $c_k$ is composed of three terms (we drop here the index $k$ to simplify the notation):

$$c = \lambda_\gamma \cdot c^\gamma + \lambda_l \cdot c^l + \lambda_t c^t \tag{5.4.42}$$

where $c^\gamma$ is the cost due to unbalancing, $c^t$ is the transition cost and $c^l$ is the cost due to the loss of vehicles ($\lambda_\gamma$, $\lambda_t$ and $\lambda_l$ are positive constants).

In a first approach we define $c^\gamma$ as:

$$c^\gamma = \|\boldsymbol{\gamma}^r\|_2^2 = \boldsymbol{\gamma}^{r\prime} \cdot \boldsymbol{\gamma}^r, \tag{5.4.43}$$

where $\boldsymbol{\gamma}^r$ is the performance measure vector after reallocation and can be calculated as:

$$\boldsymbol{\gamma}^r = \boldsymbol{\gamma} - \boldsymbol{\mathcal{A}} \cdot \boldsymbol{\mathcal{N}} \cdot \mathbf{1}, \tag{5.4.44}$$

Note that $\boldsymbol{\gamma}$ is the performance vector before reallocation. We choose a quadratic cost function instead of a linear one due to the fact that we want to prioritize re-allocations among teams with the highest deviations from the plan. In this way we are minimizing not only the mean, but also the maximum deviation of the performance measures of teams.

**Example 5.10.** *Let us get the expression for $\boldsymbol{\gamma}^r$ for the example 5.8:*

$$\begin{aligned}
\boldsymbol{\gamma}^r &= \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix} - \\
&- \begin{bmatrix} 1 & 1 & -1 & 0 & -1 & 0 \\ -1 & 0 & 1 & 1 & 0 & -1 \\ 0 & -1 & 0 & -1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{3,2} & a_{3,3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{5,4} & a_{5,5} & a_{5,6} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \\
&= \begin{bmatrix} -2 + a_{3,2} + a_{3,3} + a_{5,4} + a_{5,5} + a_{5,6} \\ 1 - a_{3,2} - a_{3,3} \\ 2 - a_{5,4} - a_{5,5} - a_{5,6} \end{bmatrix}
\end{aligned} \tag{5.4.45}$$

*The expression for $c^\gamma$ will be:*

$$c^\gamma = (-2 + a_{3,2} + a_{3,3} + a_{5,4} + a_{5,5} + a_{5,6})^2 + (1 - a_{3,2} - a_{3,3})^2 + (2 - a_{5,4} - a_{5,5} - a_{5,6})^2 \tag{5.4.46}$$

Using the $c^\gamma$ defined in equation 5.4.44 can lead to some "misleading" solutions. Let us take the following example with three teams and 6 vehicles. Assume that for a certain stage we have the plan $\boldsymbol{p}' = [2\,1\,1]$ and the number of vehicles per team is $(\mathbf{Z} \cdot \mathbf{1})' = [1\,2\,3]$. This will give us the performance measure vector $\boldsymbol{\gamma}' = [-1\,1\,2]$. If no reallocations are done, the cost using equation 5.4.44 will be $c^\gamma = 6$. Consider now the case where the third team has lost a vehicle. The performance measure vector will now be $\boldsymbol{\gamma}' = [-1\,1\,1]$ and the corresponding cost is $c^\gamma = 3$. In fact loosing a vehicle in this situation led to an apparently more balanced situation, but this is not the desired behavior. In order to penalize such situations, we can redefine $c^\gamma$ as:

$$c^\gamma = \begin{cases} \boldsymbol{\gamma}^{r\prime} \cdot \boldsymbol{\gamma}^r, & \text{if } \mathbf{1}' \cdot \mathbf{Z} \cdot \mathbf{1} = N_v; \\ \max\left\{\boldsymbol{\gamma}^{r\prime} \cdot \boldsymbol{\gamma}^r, \max_{\boldsymbol{\mathcal{X}} \in X_v} c^\gamma(x)\right\}, & \text{if } \mathbf{1}' \cdot \mathbf{Z} \cdot \mathbf{1} < N_v. \end{cases} \tag{5.4.47}$$

where $X_v$ is the set of states without loss of vehicles (i.e. the number of vehicles is equal to the initial number of vehicles $N_v$ or more precisely if $\mathbf{1}' \cdot \mathbf{Z} \cdot \mathbf{1} = N_v$). The meaning of equation 5.4.47 is that if a state has less vehicles than the initial number of vehicles $N_v$, then the cost $c^\gamma$ will never be lower then the highest $c^\gamma$ of the set of states without loss of vehicles ($X_v$). Note that $c^\gamma$ is a recursive function due to the fact that $c^\gamma$ has to be calculated first for all $\boldsymbol{\mathcal{X}} \in X_v$ in order to calculate the cost for the remaining states.

This $c^\gamma$ gives rises to a fair solution. However, it may lead to very conservative solutions which may affect the balance among teams.

Another way to solve this undesirable behavior is to maintain $c^\gamma = \boldsymbol{\gamma}^{r\prime} \cdot \boldsymbol{\gamma}^r$ and introduce a term in the cost function to penalize states where vehicles are lost. This term is defined as:

$$c^l = N_v - \mathbf{1}' \cdot \mathbf{Z} \cdot \mathbf{1} + \mathbf{1}' \cdot \mathbf{W} \cdot \mathbf{1}. \tag{5.4.48}$$

where $c^l$ is the cost due to the loss of vehicles, $\mathbf{1}' \cdot \mathbf{Z} \cdot \mathbf{1}$ gives the number of vehicles in allocation matrix $\mathbf{Z}$, and $\mathbf{1}' \cdot \mathbf{W} \cdot \mathbf{1}$ gives the number of vehicles lost during the corresponding stage. The relevance of this term in the problem is controlled by the multiplier $\lambda_l$.

Finally we add a term that reflects the costs due to transitions of vehicles, $c^t$:

$$c^t = \boldsymbol{c}' \cdot \boldsymbol{\mathcal{A}} \cdot \mathbf{1} \tag{5.4.49}$$

where $\boldsymbol{c} \in \mathbb{R}_+^{|T_r|}$ is the vector of transitions cost (eg. distance among teams, fuel consumption, etc.).

### 5.4.6 Problem statement

We are now able to model the dynamic reallocation problem. We model it as a finite horizon Markov Decision Process (MDP). The system dynamics are given by the state update function $F_k : X_k \times A_k \times \Omega_k \rightarrow X_{k+1}$:

$$\boldsymbol{\mathcal{X}}_{k+1} = F_k(\boldsymbol{\mathcal{X}}_k, \boldsymbol{\mathcal{A}}_k, \mathbf{W}_k) \tag{5.4.50}$$

where $\boldsymbol{\mathcal{X}}_k \in X_k$ is the state at stage $k$. The state may include in addition to the allocation matrix, the plan, the fuel reserves, available weapons, etc.:

$$\boldsymbol{\mathcal{X}}_k = \begin{bmatrix} \mathbf{Z}_k \\ \boldsymbol{p}_k \\ \boldsymbol{f}_{rk} \\ \cdots \end{bmatrix} \tag{5.4.51}$$

The update function 5.4.50 is expressed by equation 5.4.14 for the simpler case when the state consists only of $\mathbf{Z}$.

The dynamic reallocation control problem can be viewed as one of minimizing the following expected cost:

$$\text{minimize} \underset{\omega_k \in \Omega_k}{\mathbf{E}} \left( c_N(\boldsymbol{\mathcal{X}}_N) + \sum_{k=1}^{N-1} c_k(\boldsymbol{\mathcal{X}}_k, \mu(\boldsymbol{\mathcal{X}}_k), \omega_k) \right) \tag{5.4.52}$$

where $\mu_k : X_k \rightarrow A_k$ is a function that maps the state into a possible control action, i.e. a feedback control law:

$$\mu(\boldsymbol{\mathcal{X}}_k) = \boldsymbol{\mathcal{A}}_k \tag{5.4.53}$$

$$\begin{aligned}
&\text{subject to:} \\
&\text{Conservation constraint} && \boldsymbol{\mathcal{A}}'_k \cdot \mathbf{1} \leq \mathbf{1} \\
&\text{Vehicles constraint: Fuel} && \boldsymbol{\mathcal{A}}'_k \cdot \boldsymbol{c}_{fk} \leq \boldsymbol{f}_{rk} \\
& && \cdots \\
&\text{Operators constraint} && \boldsymbol{\mathcal{A}}'_k \cdot \mathbf{1} \leq \boldsymbol{b}_v \\
&\text{Suppliers constraint} && \boldsymbol{\mathcal{N}}_{sk} \cdot \boldsymbol{\mathcal{A}}_k \cdot \mathbf{1} \leq \boldsymbol{\gamma}^s_k + \boldsymbol{b}^s_k \\
&\text{Demanders constraint} && \boldsymbol{\mathcal{N}}_{dk} \cdot \boldsymbol{\mathcal{A}}_k \cdot \mathbf{1} \leq -\boldsymbol{\gamma}^d_k + \boldsymbol{b}^d_k
\end{aligned}$$

$\pi = \{\mu_1, \mu_2, ..., \mu_{N-1}\}$ is a control policy for the problem, $c_k : X_k \times A_k \rightarrow \mathbb{R}^+$ is the cost at stage $k$, $c_N : X_N \rightarrow \mathbb{R}^+$ is the cost at the final stage $k = N$, $\omega_k \in \Omega_k$ is the uncertainty at stage $k$ (destruction of vehicles due to SAMs and pop-up threats).

The problem stated in 5.4.52 is a MDP and can be solved using the Dynamic Programming framework. In section 3.4 we present an overview of the DP framework. In chapter 6 we present our approach to solve the reallocation problem using DP and allowing mixed-initiative interactions.

### 5.4.7 Existence of solution

After modeling an optimization problem, it is important to check if a solution of the problem exists. In the case of the dynamic reallocation problem stated in equation 5.4.52 we have a discrete-time, finite-horizon MDP. In such cases we just have to ensure that the cost function is bounded below if the problem is a minimization or bounded above if

the problem is a maximization. Thus, we just have to ensure that the expected cost of equation 5.4.52 is bounded below which is actually true because all the contributions of the cost function $c$ stated in equation 5.4.42 are non-negative (the Lagrange multipliers ($\lambda_\gamma$, $\lambda_l$ e $\lambda_t$) are also non-negative real numbers).

### 5.4.8 Mixed initiative interactions

As we have seen so far, the operator can interact with the system by the adjustment of several parameters of the system. Next we enumerate the different aspects of human intervention in our framework:

1. Allocation planning procedure:
   (a) The operator is able to re-plan if some new estimate of the world is provided (e.g. update on SAMs lethality or new pop-up threats based on some new intelligence);
   (b) The operator is allowed to choose the final probability of success, $P_f$, and also to change the demand of vehicles in each stage of mission by changing the plan matrix $\mathbf{P}$;
2. Operators constraint ($\boldsymbol{\mathcal{A}}'_k \cdot \mathbf{1} \leq \boldsymbol{b}_v$): with this constraint the operator has the ability to permanently allocate vehicles to a given team;
3. Boundaries on the demand and supply constraints ($\boldsymbol{b}^s$ and $\boldsymbol{b}^d$): With these boundaries the operator can relax or stiffen the corresponding constraints (equations 5.4.29 and 5.4.30), e.g. a supply team that should deliver no more than 4 vehicles could deliver one more if the corresponding element in $\boldsymbol{b}^s$ is set to 1;
4. The weights of the cost function ($\lambda_\gamma$, $\lambda_l$ and $\lambda_t$): With access to the cost function multipliers the operator can weight the cost parameters according to their importance. For example, if the operator wants to penalize more the situations where it is possible to lose vehicles, he may increase $\lambda_t$;
5. Transition probabilities function $P_k(\mathcal{X}_{k+1}|\mathcal{X}_k, \mathcal{A}_k)$: the operator can update the transition probabilities function with information received from an intelligence facility, such as an Airborne Early Warning and Control System (AWACS).

The degrees of freedom in the items 1b, 2, 3 and 4 should be defined by the operator during mission planning and readjusted in run time only when it is really necessary. In fact, these parameters model the way the operator perceives risk and uncertainty and should be changed only based on experience. Items 1a and 5 are more critical due to the fact that they are related to changes in the world occurring during execution.

Let us formalize the intervention of the operator. This is done with the help of the vector $\boldsymbol{\xi} \in \Xi$. The dimension of this vector is equal to the number of degrees of freedom given to the operator. The space $\Xi$ is the set with all possible combinations of values of the degrees of freedom. In our reallocation problem, $\boldsymbol{\xi}$ contains the following operator's degrees of freedom:

$$\boldsymbol{\xi} = \begin{bmatrix} \mathbf{P} \\ \boldsymbol{b}_v \\ \boldsymbol{b}_s \\ \boldsymbol{b}_d \\ \lambda_\gamma \\ \lambda_l \\ \lambda_t \\ P_k(\mathcal{X}_{k+1}|\mathcal{X}_k, \mathcal{A}_k) \end{bmatrix} \tag{5.4.54}$$

Consider that $\boldsymbol{\psi}_k \in \Psi$ is an informal vector that represents the worlds features that are not considered in the mathematic model, but are available to the operator at stage $k$. The structure of this vector could be based on the subscription of some information provided by other systems and by the operator's perception of the state of the world. An example follows:

$$\boldsymbol{\psi} = \begin{bmatrix} weather\_conditions \in \{light, accetable, severe\} \\ night\_operation \in \{0, 1\} \\ AWACS\_intelligence \in \{0, 1\} \\ troops\_on\_field \in \{0, 1\} \\ civil\_facilities \in \{0, 1\} \\ \cdots \end{bmatrix} \tag{5.4.55}$$

In chapter 6 we will see when and how often the operator is able to perform updates on the system's parameters. We formalize the updates with the partial function $\phi : \Xi \times \Psi \rightarrow \Xi$ that models the way the operator may change the parameters in $\boldsymbol{\xi}$. Thus, $\phi$ maps at a stage $i$ the parameters $\boldsymbol{\xi}_i$ and the available information $\boldsymbol{\psi}_i$ into a new vector $\boldsymbol{\xi}_{i+1}$:

$$\boldsymbol{\xi}_{i+1} = \phi(\boldsymbol{\xi}_i, \boldsymbol{\psi}_i). \tag{5.4.56}$$

Note that the index $i$ in the above equation represents the stages of operator's interaction and is different from the index $k$ used so far which represents the stages of tasks.

# Chapter 6

# Approach

In this chapter we present our approach to solve the dynamic reallocations problem presented on chapter 5. Our solution implements a hierarchical control architecture. The controllers within this architecture allow for mixed initiative interactions.

The dynamic reallocation problem is of combinatorial nature. We present some techniques to evaluate and reduce the complexity of the problem. We also present a procedure to manage complexity using mixed-initiative interactions.

## 6.1 Hierarchical decomposition

Let us now introduce in figure 6.1 the hierarchical control architecture adopted from [29]. The "UCAV Controller" is in charge of controlling the activities of each UCAV. This is achieved through switching among prototypical maneuvers such as navigation maneuvers, strike maneuvers, jamming procedures, and loiter maneuvers. There is one

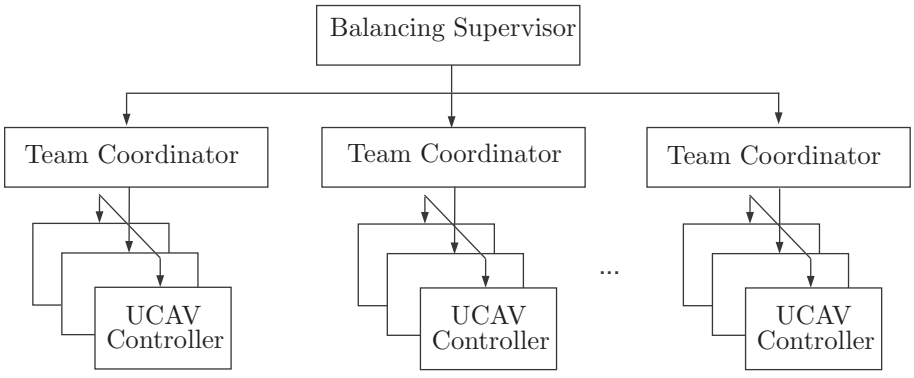



Figure 6.1: Control hierarchy.

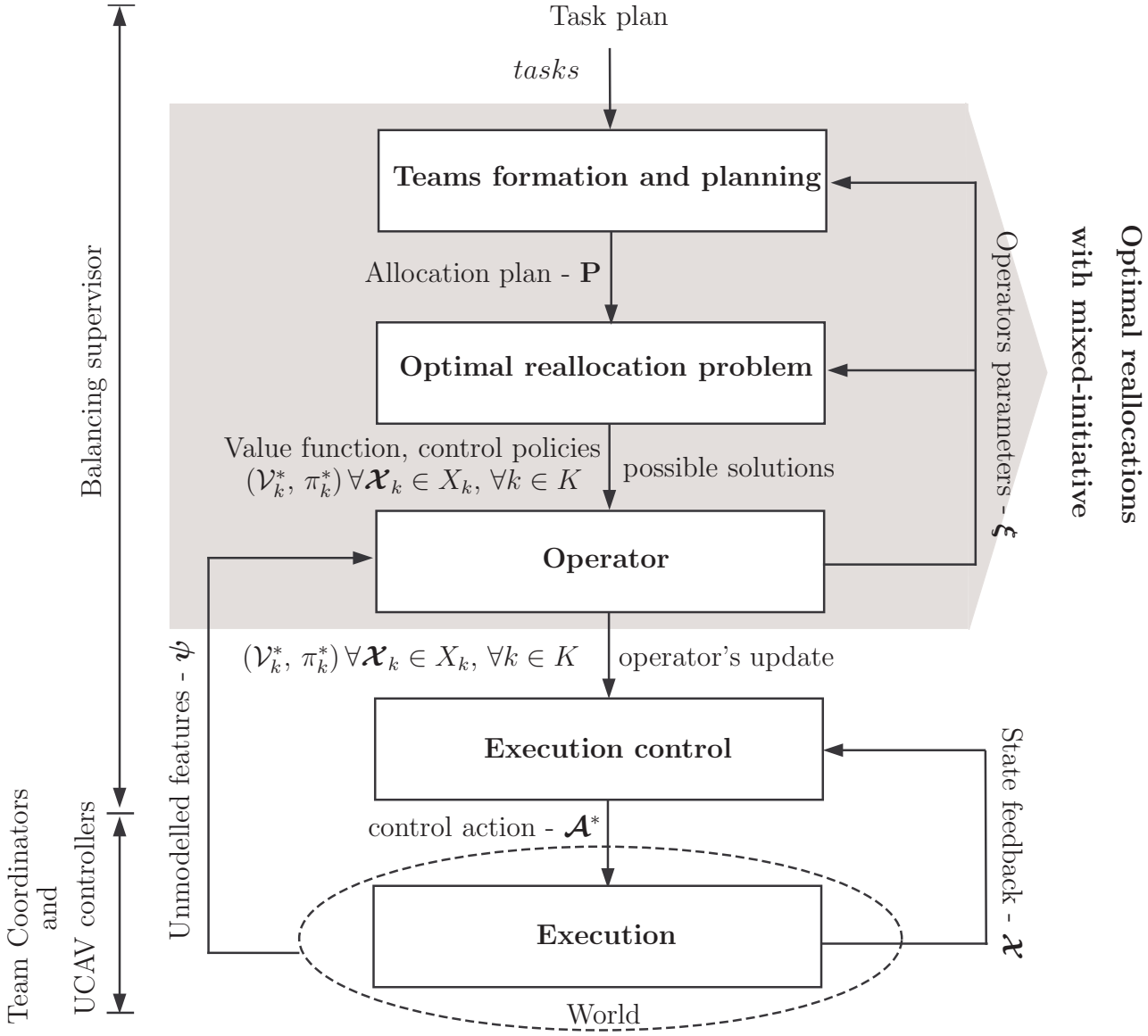



Figure 6.2: Balancing supervisor level in more detail.

UCAV controller per UCAV. The "Team Coordinator" supervises the execution of tasks by a team of UCAVs. It is responsible to instantiate the attack controller presented at section 5.3.3. The "Balancing Supervisor", which is at the core of this work, is responsible for all reallocations of vehicles among teams. Figure 6.2 depicts with more detail the balancing supervisor with special emphasis on its layered structure. The first module "Teams formation and planning" concerns the allocation of teams to tasks described in section 5.3.2 and the allocation planning procedure presented on section 5.3.4. This level assumes that a task plan is provided by a task plan procedure executed by some high-level decision maker. The module "Optimal reallocation problem" solves the optimal reallocation control problem formalized in equation 5.4.52. To solve this problem we use

the Stochastic Dynamic Programming framework (see section 3.4). The DP algorithm evaluates the optimal control policy for the whole state-space at each stage. The optimal control policies and the corresponding value function are delivered to the "Operator" module for approval. In the "Operator" module, the operator is allowed to accept the control policies, to change some parameters, and to ask for a new re-calculation. The degrees of freedom of the operator are enumerated in section 5.4.8. After the operator accepts the control policies from the optimal control problem, the results are loaded in a look-up table, i.e. a list that matches the possible states at each stage to the corresponding optimal decisions. The modules "Teams formation and planning", "Optimal reallocation problem", and "Operator", perform what we call the "Optimal reallocations with mixed-initiative". The execution of reallocations is performed by the lower levels of the control hierarchy of figure 6.1, namely the "Team Coordinators" and the "UCAV Controllers".

## 6.2 Solving the execution control problem using the DP framework

We modeled the execution control problem in section 5.4.6 as a MDP. Let us restate the problem presented in equation 5.4.52 and equation 5.4.53:

$$\mathcal{V}_k^*(\boldsymbol{\mathcal{X}}_k) = \min_{\mu_k} \mathop{\mathbf{E}}_{\omega_k \in \Omega_k} \left( c_N(\boldsymbol{\mathcal{X}}_N) + \sum_{k=1}^{N-1} c_k(\boldsymbol{\mathcal{X}}_k, \mu(\boldsymbol{\mathcal{X}}_k), \omega_k) \right) \quad (6.2.1)$$

subject to:

$$\mu(\boldsymbol{\mathcal{X}}_k) = \boldsymbol{\mathcal{A}}_k \quad (6.2.2)$$

Conservation constraint
$$\boldsymbol{\mathcal{A}}'_k \cdot \mathbf{1} \leq \mathbf{1} \quad (6.2.3)$$

Vehicles constraint: Fuel
$$\boldsymbol{\mathcal{A}}'_k \cdot \boldsymbol{c}_{fk} \leq \boldsymbol{f}_{rk} \quad (6.2.4)$$

$$\ldots$$

Operators constraint
$$\boldsymbol{\mathcal{A}}'_k \cdot \mathbf{1} \leq \boldsymbol{b}_v \quad (6.2.5)$$

Suppliers constraint
$$\boldsymbol{\mathcal{N}}_{sk} \cdot \boldsymbol{\mathcal{A}}_k \cdot \mathbf{1} \leq \boldsymbol{\gamma}_k^s + \boldsymbol{b}_k^s \quad (6.2.6)$$

Demanders constraint
$$\boldsymbol{\mathcal{N}}_{dk} \cdot \boldsymbol{\mathcal{A}}_k \cdot \mathbf{1} \leq -\boldsymbol{\gamma}_k^d + \boldsymbol{b}_k^d \quad (6.2.7)$$

### 6.2.1 Optimal control strategy

The value function stated in equation 6.2.1 can be solved using the Dynamic Programming framework previous explained in section 3.4. For the sake of clarity let us state here the Bellman's Recursion, previously presented in equation 3.4.2 but now adapted to our problem's notation:

$$\mathcal{V}_k^*(\boldsymbol{\mathcal{X}}_k) = \min_{\mu_k} \left( c_k(\boldsymbol{\mathcal{X}}_k, \mu_k(\boldsymbol{\mathcal{X}}_k), \boldsymbol{\omega}_k) + \sum_{i=1}^{n} P_k(\boldsymbol{\mathcal{X}}_i | \boldsymbol{\mathcal{X}}_k, \mu_k(\boldsymbol{\mathcal{X}}_k)) \cdot \mathcal{V}_{k+1}^*(\boldsymbol{\mathcal{X}}_i) \right), \forall \boldsymbol{\mathcal{X}}_k \in X_k \tag{6.2.8}$$

The most relevant drawback on using the Bellman's recursion to solve our execution control problem is complexity. The problem faces a curse of dimensionality. This issue has to be handled carefully otherwise the implemented algorithm may not produce the desired results in a feasible time. We discuss this problem with more detail next.

### 6.2.2 Curse of dimensionality

The curse of dimensionality (also known as state explosion) is defined by Bellman [38] as [sic]: "...the exponentially grow of running time of algorithms with the dimension of state space". This is a considerable obstacle for the use of Bellman recursion of equation 6.2.8 due to the fact that all $\boldsymbol{\mathcal{X}}_k \in X_k$ have to be evaluated in each step $k$. Remind that the allocation matrix $\mathbf{Z}$ is part of the state $\boldsymbol{\mathcal{X}}$ and the procedure to achieve all possible matrices $\mathbf{Z}$ is a highly combinatorial task. Consider the following example.

**Example 6.1.** *Let us suppose that, at a certain stage $k$, the state $\boldsymbol{\mathcal{X}}_k$ is composed by the allocation matrix $\mathbf{Z}_k$, the vector $\boldsymbol{p}_k \in \mathbb{Z}^{N_t}$ which is the $k^{th}$ row of matrix plan $\mathbf{P}$, and the fuel reserves $f \in \mathbb{R}^{N_v}$ (in fact the state will contain even more information like the position of vehicles but for the sake of simplicity we will just consider these elements).*

*Each column of matrix $\mathbf{Z}$ can take $N_t + 1$ different configurations: vehicle allocated in one of the $N_t$ teams or destroyed/neutralized/out of assets. The number of different possible $\mathbf{Z}$ matrices (possible allocations of $N_v$ vehicles in $N_t$ teams) will thus be:*

$$N_z = (N_t + 1)^{N_v} \tag{6.2.9}$$

*Let us consider that for each team $i$, $p_k(i) \leq \lfloor N_v/N_t \rfloor$, i.e. the plan for each team, is at most, the immediately integer below the total number of vehicles divided by the total*

*number of teams. Thus, the number of different plans for a team is* $\lfloor N_v/N_t \rfloor + 1$ *(e.g. if for certain team* $i$*,* $p_k(i) \leq 3$ *then we may have 4 plans:* $p_k(i) \in \{3, 2, 1, 0\}$*). This leads us to the following number of possible combinations of* $\boldsymbol{p}_k$ *vectors:*

$$N_p = (\lfloor N_v/N_t \rfloor + 1) \times N_t \tag{6.2.10}$$

*Consider now that the initial fuel in each vehicle is* $F$ *and that we assume discrete values of fuel with increments of* $\Delta f$*. We also assume that each vehicle could have* $\lfloor F/\Delta f \rfloor + 1$ *different levels of fuel (including the zero level). Thus, the number of possible combinations of vector* $f$ *is:*

$$N_f = (\lfloor F/\Delta f \rfloor + 1) \times N_v \tag{6.2.11}$$

*For the present case, the cardinality of state space will be:*

$$|X| = N_z \times N_p \times N_f \tag{6.2.12}$$

*Consider that* $N_t = 3$*,* $F = 100$*, and* $\Delta f = 1$*. In figure 6.3 we present the cardinality of state space for a number of vehicles ranging from* 1 *to* 100 *(both domain and range are presented in logarithmic scales). Analyzing figure 6.3 we can notice the huge growth of*

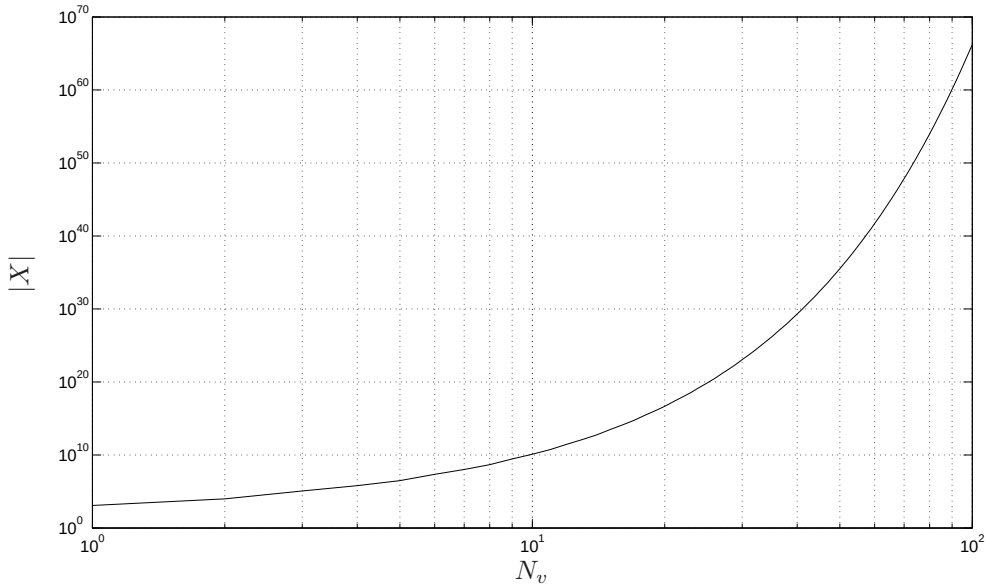



Figure 6.3: $|X|$ for $N_t = 3$, $F = 100$, and $\Delta f = 1$ as function of $N_v$.

*state space.*

For the sake of simplicity we didn't include in the state vector the position of vehicles, as well as, other kind of vehicles' characteristics (e.g. weapons reserves). Thus, in a real scenario the cardinality of the state space will be even greater.

One way to reduce the dimension of the state space is to aggregate state variables. In section 3.4.3 we have explored several aggregation approaches. Let us next consider some examples using hard aggregation.

**Example 6.2.** *Let us consider the example 6.1 but now with two hard aggregations: fuel can only take one of three values (for example one can label such values as "high", "medium", "low") and matrix* $\mathbf{P} \in \{\ldots -4, -2, 0, 2, 4 \ldots\}^{N \times |T|}$ *instead of* $\mathbf{P} \in \mathbb{Z}^{N \times |T|}$*, i.e.* $\mathbf{P}$ *takes only even values. In practice, if a team demands (or delivers) an odd number of vehicles, we consider the integer that precedes it. In figure 6.4 it is visible a decrease*

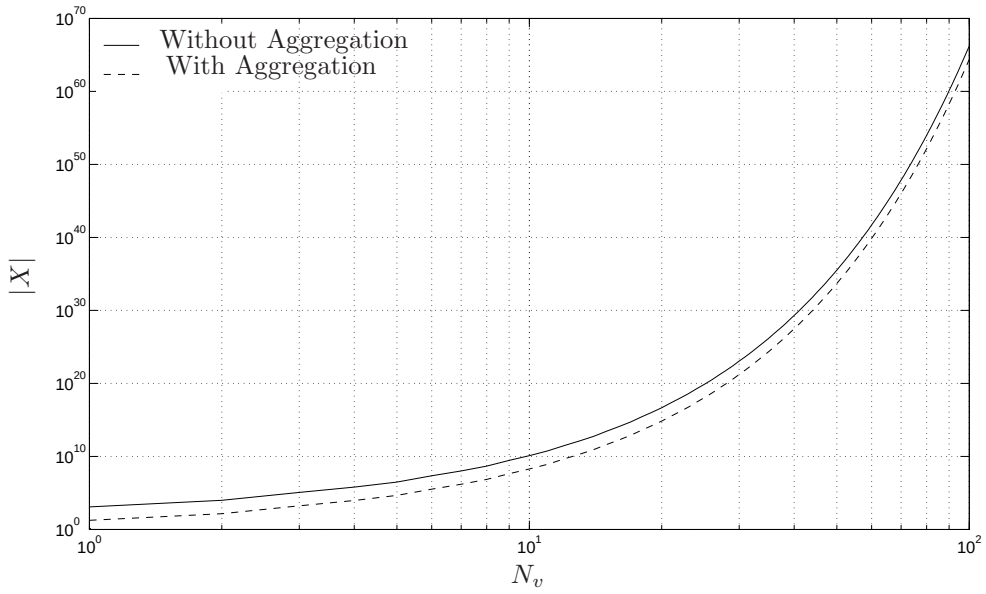



Figure 6.4: $|X|$ of example 6.1 with aggregation in fuel and plan.

*on the number of states. For* $N_v = 10$ *the reduction is about two orders of magnitude. However, the curse of dimensionality remains, mainly due to all the possible combinations in the matrix* $\mathbf{Z}$*.*

The allocations of vehicles to teams are, in fact, the main source of the state explosion. Let us now consider an aggregation on $Z$.

**Example 6.3.** *Instead of consider the allocation of vehicles to teams we consider an allocation of a number of vehicles to teams (e.g. team 1 has 5 vehicles, team 2 has 3*

*vehicles...). Using our mathematical notation, this is the same of using* $\mathbf{Z} \cdot \mathbf{1}$ *in the state instead of* $\mathbf{Z}$*. In this case the possible allocations are:*

$$N_z = (N_v + 1) \times (N_t - 1) \tag{6.2.13}$$

*Using this aggregation we are giving up on considering the specific attributes of each UCAV, i.e. their heterogeneity. Thus,* $N_f = 1$*. The number of possible plans (*$N_p$*) does not change from the previous example.*

*Let us apply this aggregation technique to example 6.1 and compare the cardinality of state space to the one obtained in example 6.2. Analyzing the figure 6.5 we can notice a high*

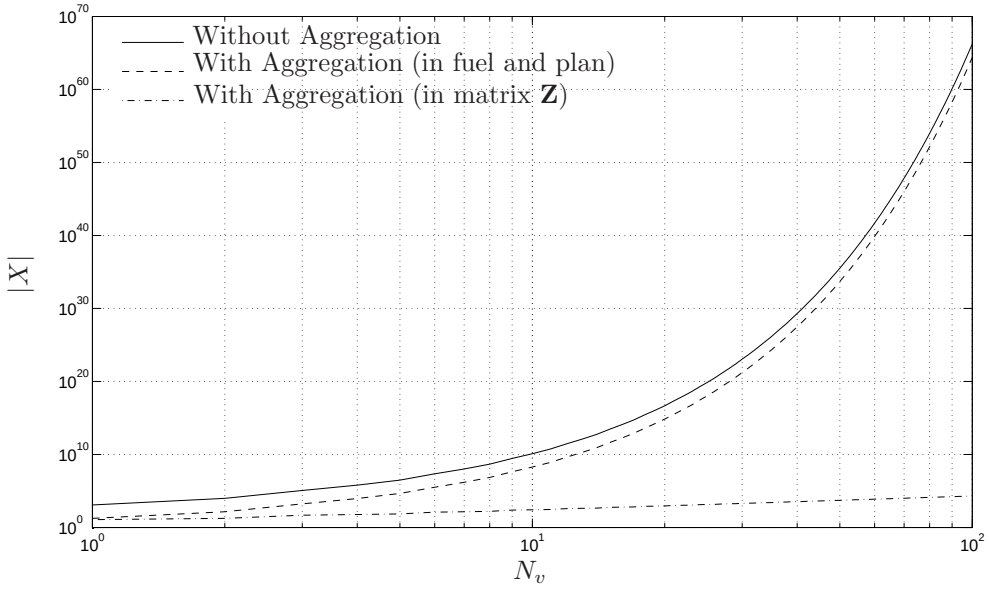



Figure 6.5: Cardinality of state space for example 6.1 with two aggregation techniques.

*reduction on the state space cardinality. However, like it was mentioned above, with this aggregation the vehicles characteristics such as ID, fuel reserves, and weapon reserves, are not taken into account in the reallocation problem.*

### 6.2.3 Greedy control strategy

Due to the fact that the Bellman's recursion implies evaluation over the entire planning horizon, sometimes it may be worth to solve the problem by just considering the next stage rather the whole horizon. Such control strategy will decrease the problem's complexity comparing to the Bellman's recursion. However, it is evident that optimality is

not guaranteed. In literature, this kind of control strategies are called "greedy". We implemented a greedy algorithm to solve the reallocation control problem.

Keep in mind equation 3.4.21 where we presented a greedy version for Bellman's recursion. Adapting to our problem's notation we have:

$$\mathcal{V}^*_{gk}(\boldsymbol{\mathcal{X}}_k) = \min_{\mu_k} \mathop{\mathbf{E}}_{\boldsymbol{\omega}_k \in \Omega_k} \left( c_k(\boldsymbol{\mathcal{X}}_k, \mu(\boldsymbol{\mathcal{X}}_k), \boldsymbol{\omega}_k) + c_{k+1}(\boldsymbol{\mathcal{X}}_{k+1}) \right). \tag{6.2.14}$$

## 6.3 Mixed-initiative for execution control

The mixed-initiative interactions aim at the use of the operator's knowledge in order to enrich the overall system and improve the safety and the performance of system. However, modeling such interactions is not a simple task. Several questions arise when one tries to model interactions within a human operator and a mathematically modeled system: what would be the degrees of freedom provided to the operator? when and how the system should request the operator's intervention? and what decision aids should the system provide to the operator in order to support his decision?

The answers to such questions are highly dependent on the problem we are solving. Let us present our approach for the dynamic reallocation problem.

### 6.3.1 Model

The model for mixed-initiative interactions is depicted in figure 6.6. Let us describe briefly the function of each block of figure 6.6. The first level, "Optimal Reallocation problem", is basically the merge of the two first levels of figure 6.2, i.e. the "teams formation and planning" and the "optimal reallocation problem" modules. This level presents the operator with: 1) value function and 2) the optimum policies for all possible states and for the remaining stages. The operator's level is the bottom grey box of figure 6.6. Besides the value function and the optimum policies, there are other inputs to this level, namely, the problem's complexity and a vector with the unmodelled world features ($\boldsymbol{\psi}$), which are introduced on the fly. The outputs are the approved value function and optimum control policies ($(\mathcal{V}^*_k, \pi^*_k) \forall \boldsymbol{\mathcal{X}}_k \in X_k$) and the vector with operator's degrees of freedom ($\boldsymbol{\xi}$). The prior and the new value function and the respective control policies are stored. A decision

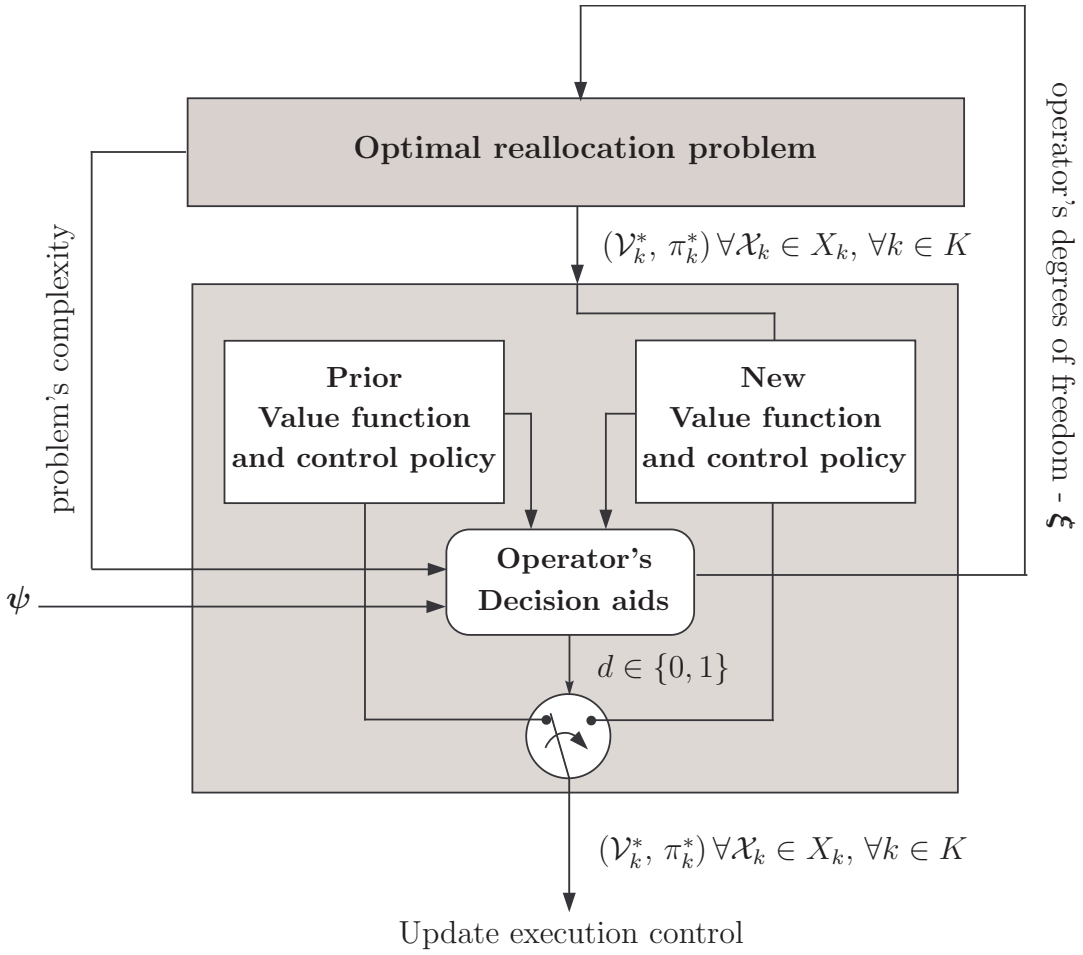



Figure 6.6: Mixed-initiative interactions.

aid helps the operator to choose if he wants to keep the prior value function and control policies ($d = 0$), or ask for recalculations until he is satisfied and accept the new results ($d = 1$).

Next we discuss the constraints for scheduling and trigger mixed initiative interactions.

### 6.3.2 Constraints for scheduling interventions

In our framework the operator is able to change the execution control strategy (update the execution control at the bottom of figure 6.6) during periods of time when teams are moving from one attack to another. During the attacks updates are not allowed. We impose such constraint by introducing before each attack a guard on time during which any kind of reallocations are forbidden.

Let us take a simple example.

**Example 6.4.** *Consider the following example where the operator decides to change the current plan twice: before the first and the second attacks. The result is depicted in the time line of figure 6.7. In the example of figure 6.7, the operator interacts with the system*

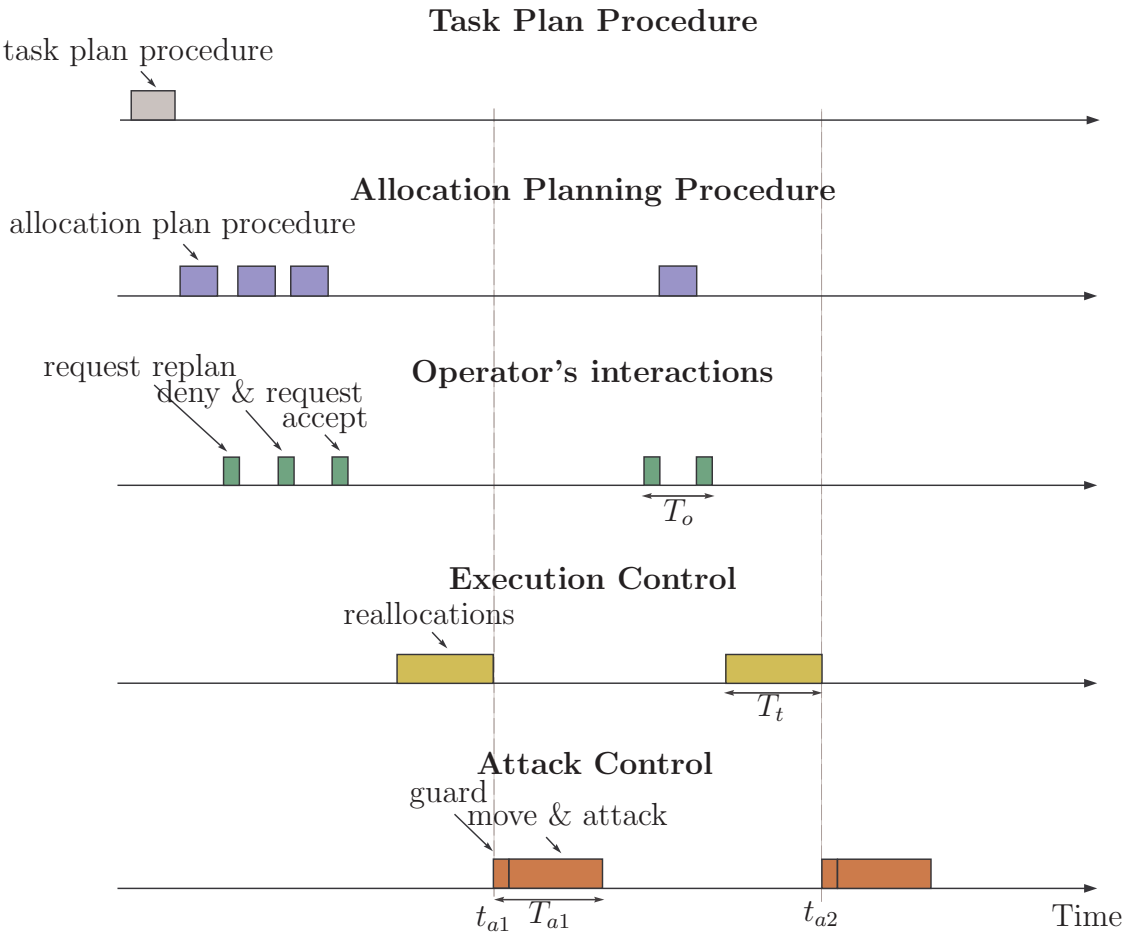



Figure 6.7: Example of a time line with the operator's interactions.

*after the first plan is delivered by the allocation planning procedure. The operator invokes the allocation planning procedure until he is satisfied with the result (in this case two interactions). He repeats the procedure before the second attack.*

Consider again figure 6.7. Assuming that the attacks on stage $n$ have to start on the deadline $t_{a\,n}$, the number of interactions that the operator could execute between attacks $n$ and $n+1$ is:

$$n_o(n) = \frac{t_{a\,n+1} - t_{a\,n} - T_{a\,n} - T_t}{T_o} \tag{6.3.1}$$

where $T_{a\,n}$ is the duration of the attacks at stage $n$, $T_t$ is the maximum time allowed for transitions, and $T_o$ is the maximum time allowed between asking for replan and to accept the new plan.

Note that a time constraint should be included in the execution control problem in order to prevent vehicles to violate deadlines during a reallocation. Thus, the problem should be reformulated as a continuous-time or discrete-time problem and the system should be modeled as a hybrid system in order to include these hybrid dynamics.

### 6.3.3 Decision aids

We assume that the operator's intervention is based on emergent information that is not accessible to the system (remind that such information is stated on vector $\boldsymbol{\psi}$). Thus, it is not possible for the system to evaluate the goodness of the operator's interventions. Still, assuming that it is desirable that the system runs as initially planned, the system can provide the operator with information on how it would behave under a different strategy or plan. One way to do that is by evaluating the changes in the value function.

#### 6.3.3.1 Variations on the value function

The value function gives the optimum cost and obviously we want to keep it as small as possible. The degrees of freedom for the operator are modeled as vector $\boldsymbol{\xi}$ in equation 5.4.54. Changing $\boldsymbol{\xi}$, the operator is also changing the value function 3.3.8. To evaluate what are the implications of changes in $\boldsymbol{\xi}$ we introduce the function $g : \Xi \times \Xi \rightarrow \mathbb{R}$:

$$g(\boldsymbol{\xi}_{i+1}, \boldsymbol{\xi}_i) = \mathcal{V}_k^*(\boldsymbol{\mathcal{X}}_k, \boldsymbol{\xi}_{i+1}) - \mathcal{V}_k^*(\boldsymbol{\mathcal{X}}_k, \boldsymbol{\xi}_i). \tag{6.3.2}$$

where $\mathcal{V}_k^*(\boldsymbol{\mathcal{X}}_k, \boldsymbol{\xi}_i)$ is the value function evaluated at the current state $\mathcal{X}_k$ and with the current parameters $\boldsymbol{\xi}_i$ and $\mathcal{V}_k^*(\boldsymbol{\mathcal{X}}_k, \boldsymbol{\xi}_{i+1})$ is the value function at state $\boldsymbol{\mathcal{X}}_k$ but now with the new parameters $\boldsymbol{\xi}_{i+1}$. Remind that the index $i$ is the operator's decision stage. Keep in mind that the value function $\mathcal{V}_k^*$ may not explicitly reflect variations of the unmodelled features (vector $\boldsymbol{\psi}$) and, thus, the function $g$ is only an indication of how $\boldsymbol{\xi}$ is changing $\mathcal{V}_k^*$.

If the $\mathcal{V}_k^*$ increases due to some changes in $\boldsymbol{\xi}$, then $g > 0$ and the operator's interface should present a warning advising that the operator's new plan will incur in a higher mission's cost. The operator then may reject the new plan and maintain the previous one or accept the additional cost and perform the modifications. For example, suppose that the operator receives information that SAMs have received an upgrade on their Electronic

Counter Countermeasures (such as frequency hopping) and they are now more resilient to the UACVs jamming. This will lead to a different $p_{kill}$ of UCAVs. Facing such situation the operator updates the matrix $\mathbf{P}$ and the probability transition function and asks for a new solution for the problem. The value function will increase due to the higher risk of loosing vehicles, and thus the system provides a warning to advertise the operator.

On the other hand, if the operator's interactions lead to a reduction of magnitude of the value function this means that the mission is easier than what was initially planned. For example, suppose that the operator receives information from an AWACS claiming that a certain SAM is no longer operational. In such situations the operator must recalculate the matrix $\mathbf{P}$ by removing the SAM from the corresponding task. In this case the value function will decrease ($g < 0$) due to the lower risk of loosing vehicles, and because it is less probable that the system may need to reallocate vehicles. In such situations the operator's interface should also present a warning that the changes will lead to an "easier" mission.

The inclusion of operator in the control loop can be thought as having a new value function for the overall problem where the vector $\boldsymbol{\psi}$ is part of the state of the system and vector $\boldsymbol{\xi}$ is included in the action space. However, the calculation of such value function is performed, not only by the system, but also by the operator. This new value function can be formalized as:

$$\mathcal{V}_k^{h*}(\boldsymbol{\mathcal{X}}_k^h) = \min_{\mu_k^h} \mathop{\mathbf{E}}_{\omega_k \in \Omega_k} \left( c_N^h(\boldsymbol{\mathcal{X}}_N^h) + \sum_{k=1}^{N-1} c_k^h(\boldsymbol{\mathcal{X}}_k^h, \mu(\boldsymbol{\mathcal{X}}_k^h), \omega_k) \right) \tag{6.3.3}$$

where

$$\boldsymbol{\mathcal{X}}_k^h = \begin{bmatrix} \boldsymbol{\mathcal{X}}_k \\ \boldsymbol{\psi}_k \end{bmatrix}, \tag{6.3.4}$$

$$\boldsymbol{\mu}(\boldsymbol{\mathcal{X}}_k^h) = \begin{bmatrix} \boldsymbol{\mu}(\boldsymbol{\mathcal{X}}_k) \\ \boldsymbol{\xi}_k(\boldsymbol{\psi}_k) \end{bmatrix}, \text{ and} \tag{6.3.5}$$

$$c_k^h(\boldsymbol{\mathcal{X}}_k^h, \mu(\boldsymbol{\mathcal{X}}_k^h), \omega_k) = c_k(\boldsymbol{\mathcal{X}}_k, \mu(\boldsymbol{\mathcal{X}}_k), \omega_k) + c_k'(\boldsymbol{\psi}_k, \boldsymbol{\xi}(\boldsymbol{\mathcal{X}}_k), \omega_k). \tag{6.3.6}$$

Note that $c' : \Psi \times \Xi \times \Omega \to \mathbb{R}$ is a cost function intrinsic of the operator. In order to model such cost function one has to take into consideration human factors such as intuition, emotions, reason and motivation (for further information on modeling human factors see [69, 63, 70]). These considerations may open new directions for research.

### 6.3.4 Managing complexity

Mixed-initiative interactions could also be useful to manage the computation complexity of the overall framework. The operator may trade a decrease in performance for a decrease in the complexity of the problem, which leads to a decrease in the computational cost (and time). As discussed in section 6.2.2 this can be done with the help of aggregation techniques. Let us take the example of the aggregation in the allocation matrix presented in example 6.3. This aggregation leads to an enormous decrease on the problem's complexity, but can actually conduce to solutions far from the optimal. However, in some practical situations, this aggregation could make sense. Consider the case where the operator knows, based on past experience, that the endurance of vehicles is long enough to perform the mission even considering several reallocations and weather uncertainties. In such situations the operator can choose to do an aggregation on the fuel consumptions, and accept the decrease on the optimality of the solution for the sake of lower complexity.

It is useful for the operator to know the problem's parameters affect the computational cost of the solution. Using the parameters that affect the problem's constraints, the operator is able to "prune" the decision tree or, on the contraire, to increase the number of possible decisions. A good example are the parameters $\boldsymbol{b}^s$ and $\boldsymbol{b}^d$ of the suppliers and demanders constraints (equations 5.4.29 and 5.4.29). These two parameters influence the number of vehicles that can be delivered or demanded by teams. For example, if the operator chooses these parameters to reduce the number of vehicles that teams can supply and/or demand then he, somehow, tights the constraints and consequently decreases the possible decisions. It is trivial to conclude that the operator's constraint of equation 5.4.24 can also lead to a decrease of complexity. With such constraint, the operator can permanently allocate vehicles to teams and thus prevent several reallocations to happen.

Similarly with other mixed-initiative interactions, handling problem's complexity implies changing parameters in vector $\boldsymbol{\xi}$. To close the loop we need to provide the operator with estimations of problem's complexity. We claim that a good way to infer the problem's complexity is using the cardinality of the state-space. Like it was mentioned above, the complexity of problems relies mostly on the size of the state space. Thus, the cardinality of the state-space ($|X|$) should be added to vector $\boldsymbol{\psi}$.

Let us now summarize the degrees of freedom:

- Aggregate the vehicles characteristics to teams (e.g. aggregation on the allocation matrix **Z**);
- Perform a relaxation technique (we didn't implemented any relaxation technique though we aim at future developments the implementation of approximate DP [43, 44]);
- Use a greedy algorithm instead of DP for the whole horizon.

# Chapter 7

# Simulations

In this chapter we present the results of simulation runs where we exercise our dynamic reallocation strategy. First we briefly present the main software modules of the simulator before discussing our simulations.

## 7.1 Introduction

In order to illustrate and demonstrate the dynamic reallocation strategy we performed several simulations runs. The simulator was developed using MATLAB® 7.0. It is not our purpose to simulate the mission's environment with fine detail and thus we considered several simplifications: 1) the UCAVs are modeled as mass points; 2) the SAMs are of the same type; and 3) the probability of kill is the same for all UCAVs. The simulations are driven by events.

## 7.2 Software modules

The simulation software was developed in several modules. The most relevant are:

- Mission Initialization;
- Allocation Procedure;
- Bellman Recursion Solver;

  Reach Set Calculation;

Optimal Strategies Generator;

- Greedy Algorithm Solver;
- Mission Simulation.

Let us now briefly explain the modules' role and relevant inputs and outputs in a pseudo-code language. With this description we want to clarify the structure of the simulator. Still, the reader should be familiar with the previous chapters in order to fully understand the main role of each module.

The *mission_initialization* procedure has the functionality of creating the world. This procedure creates the sets of vehicles, the set of SAMs, defines the tasks and establishes the probability of success for vehicle $j$ to transit from team $i$ to team $l$ ($p_s(i, j, l)$) for all $j$, $i \in V$ and $l \in T$. The inputs and outputs are:

**mission_initialization**

- Inputs: $num\_vehicles$, $vehicles\_characteristics$, $num\_SAMs$, $SAMs\_characteristics$, $other\_threats$;
- Output: $vehicles\_set$, $SAMs\_set$, $prob\_transitions$, $tasks$.

where $num\_vehicles$ is the total number of available vehicles, $vehicles\_characteristics$ is the set of vehicles characteristics such as fuel reserves and probability of kill, $SAMs\_characteristics$ is the set of the characteristics of SAMs such as their position and maximum range, $other\_threats$ represents the pop-up threats that could appear during the mission, $vehicles\_set$ is the set of UCAVs, $SAMs\_set$ is the set of SAMs, $prob\_transitions$ is the set of the probabilities of success for transiting from one team to another due to the presence of pop-up threats.

The *allocation_procedure* is responsible for the initial allocation of vehicles to teams and for the definition of the minimum number of vehicles needed at each stage for each team to accomplish a desired probability of success. The *allocation_procedure* is the software module that implements the allocation procedure detailed in section 5.3.4. The inputs and outputs of this procedure are:

**allocation_procedure**

- Inputs: $tasks$, $vehicles\_set\,(V)$, $probability\_kill\,(p_{kill})$, $desired\_probability\_success\,(P_f)$;

- Output: *plan* (**P**), *initial_allocation* (**Z**).

The *bellmans_recursion* solves the optimal control problem stated in equation 5.4.6. This software module implements the execution control problem explained in section 3.4. This is done off-line The inputs and outputs of Bellman's Recursion are:

**bellmans_recursion**

- Inputs: *initial_allocation*, *operators_constraints*, *plan*, *prob_transitions*, *aggregations*;
- Output: *look_up_table*, *costs_table*.

where *operators_constraints* defines the degrees of freedom for the operator (vector $\xi$), *aggregations* is the set of aggregations that will be used on solving the problem (e.g. the aggregation in the allocation matrix), *look_up_table* is a table with the optimal control action for each state and at each stage. The *cost_table* is a table that contains the costs for each action. The Bellman's recursion has two procedures: one to calculate the reach set for all stages and the other to find the optimal control strategy regarding all the possible states determined by the reach set procedure.

The *greedy_algorithm* solves the greedy problem stated in equation 6.2.14. This greedy algorithm is explained in the section 3.4.5. This algorithm is calculated online at each stage and is an alternative on using the Bellman's Recursion. The inputs and outputs of the greedy algorithm are:

**greedy_algorithm**

- Inputs: *actual_allocation*, *operators_constraints*, *plan*, *prob_transitions*, *aggregations*, *actual_stage*;
- Output: *control_action*, *cost*.

where *actual_stage* is the current stage of execution, the *control_action* is the greedy control action and *cost* is the cost of that action.

Finally, the *mission_simulation* is the procedure that runs the overall simulation. The inputs and outputs are:

**mission_simulation**

- Inputs: *prob_transitions*, *probability_kill*, *tasks*, *control_strategy*, *aggregations*, *initial_locations*;

- Output: $targets\_suppressed$, $vehicles\_down$, $final\_allocation$, $final\_locations$.

where $control\_strategy$ is the Bellman's recursion look-up table or the greedy algorithm depending on the selected control strategy, $targets\_suppressed$ is the set of targets suppressed, $vehicles\_down$ is the set of vehicles that were eliminated or ran out of assets, $final\_allocation$ is the allocation matrix after the last stage, $initial\_locations$ are the initial locations of vehicles and $final\_locations$ are the final positions of the vehicles. In order to give a more realistic behavior to the paths of vehicles we add some uncertainty to the position of vehicles.

## 7.3 Mission scenario

Let us now take a mission scenario to exemplify the dynamic allocation of vehicles among teams.

**Example 7.1.** *Consider that* 12 *SAMs have to be suppressed. A task plan procedure divided the attacks in three tasks that will be assigned to three teams ($t_1$, $t_2$ and $t_3$). The plan specification requires that at least* 1 *UCAV of each team reaches the end of task with probability $P_f = 0.8$. $p_{kill} = 0.7$ for all UCAVs. We assume that the probability of a vehicle being shot down during transition among teams is uniform for all stages and takes the following values:*

- $p_t(t1, t2) = p_t(t2, t1) = p_t(t2, t3) = p_t(t3, t2) = 0.001$
- $p_t(t1, t3) = p_t(t3, t1) = 0.002$

*where* $p_t : \{1 \dots |T|\} \times \{1 \dots |T|\} \rightarrow \{0, 1\}$.

*The allocation planning procedure delivers the following plan:*

$$\mathbf{P} = \begin{bmatrix} 5 & 5 & 5 \\ 4 & 4 & 4 \\ 3 & 3 & 3 \\ 2 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix} \tag{7.3.1}$$

*The rows of matrix* $\mathbf{P}$ *are the mission stages and the columns corresponds to teams. Each element is the minimum number of vehicles that the corresponding team should have to accomplish the remaining stages with a probability of success $P_f$.*

*Analyzing the first row of matrix* $\mathbf{P}$ *in equation 7.3.1 we see that we need at least* 15 *UCAVs at the first stage. Let us assume that all UCAVs are allocated to team* $t_2$ *before the mission starts. Thus, the allocation matrix at stage* 1 *is:*

$$\mathbf{Z} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7.3.2}$$

*Remember that the rows of 7.3.2 correspond to teams and the columns correspond to vehicles. We perform the simulation with the aggregation* $\mathbf{Z} \cdot \mathbf{1}$*, i.e. we allocate a number of vehicles to each team regardless the characteristics of vehicles.*

*Figure 7.1 depicts the execution of mission: UCAVs are represented with a circle and their paths are represented with grey dashed lines, the red cross means that the UCAV was shot down and the SAMs are depicted as triangles surrounded by a circle that represents their range. For a more comprehensive analysis of figure 7.1 we print the elements of matrix* $\mathbf{P}$ *close to the depiction of each SAM.*

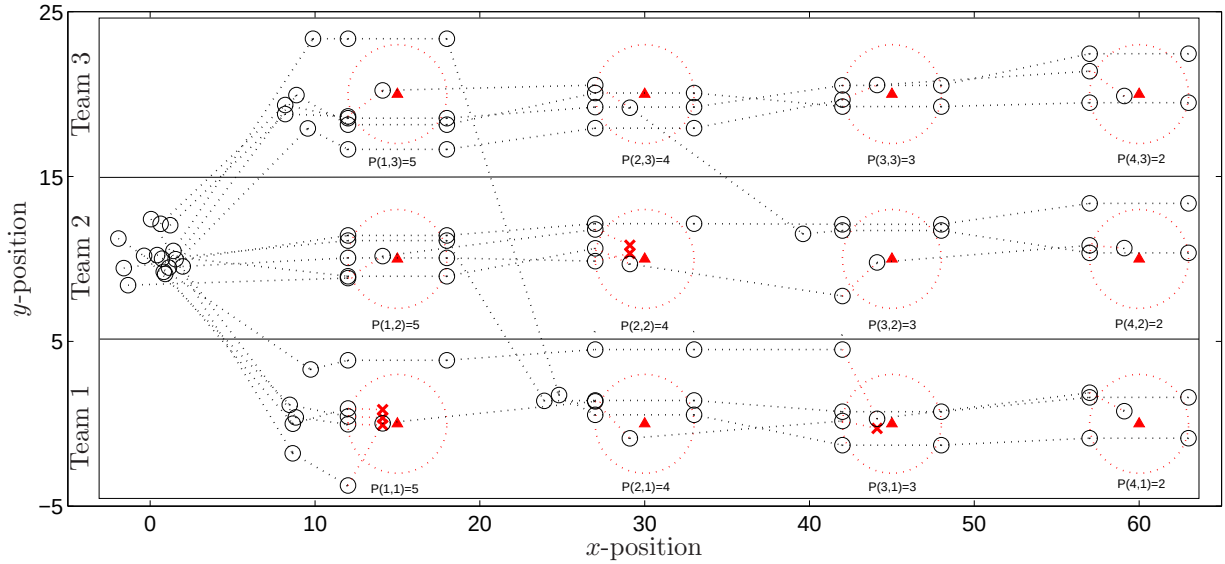



Figure 7.1: Execution of mission.

*In figures 7.2 and 7.3 we present the value function and* $\gamma^{r\prime}\gamma^{r}$ *for each stage. In figure 7.2 we see that the value function is decreasing with the number of stages. This is an expected*

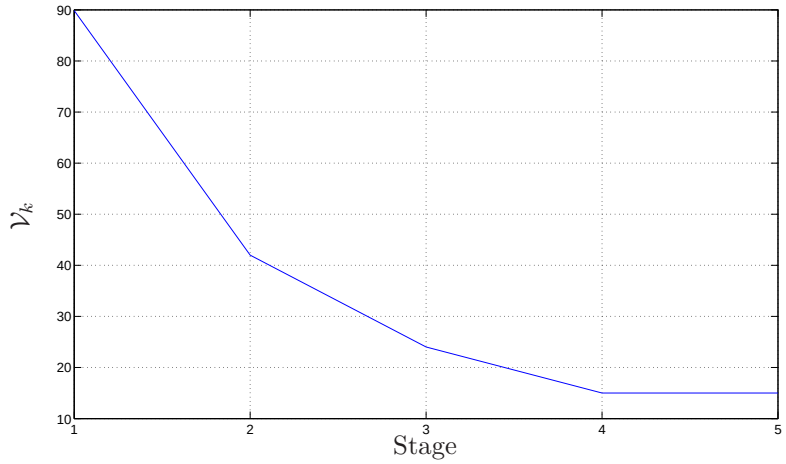



Figure 7.2: Value function for each stage.

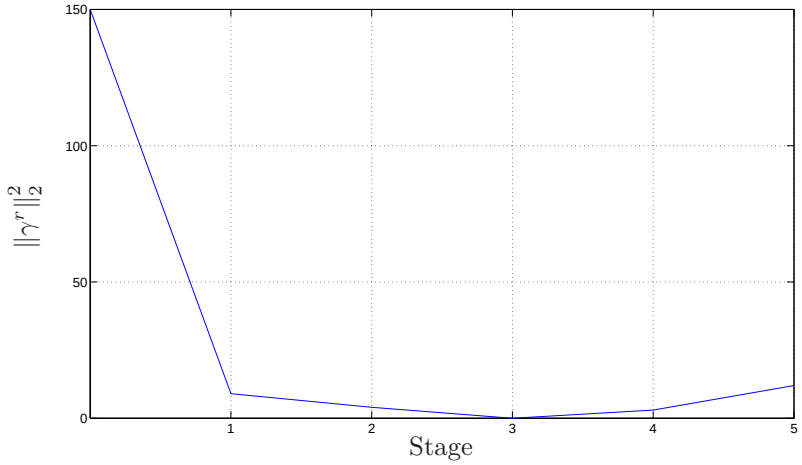



Figure 7.3: $\|\gamma^r\|_2^2$ for each stage.

*behavior due to the fact that the value function minimizes a cumulative cost (equation 3.4.2). Figure 7.3 gives an idea of the level of unbalancing and shows the expected trend. The slightly increase of* $\gamma^{r\prime}\gamma^r$ *in the last stages is due to the fact that the mission ends with more vehicles then expected. Note that the domain of figure 7.3 starts at stage* 0 *rather then at stage* 1 *like in figure 7.2. Stage* 0 *is the stage where the plan is already known, but vehicles are not allocated to their teams. We chose to start with all vehicles allocated to team* 2 *at stage* 0 *to emulate the behavior where all vehicles are at an air base before the mission starts. Thus, at stage* 0 *we have* $\gamma'\gamma$ *rather that* $\gamma^{r\prime}\gamma^r$ *because no reallocation was done yet.*

# Chapter 8

# Conclusions

## 8.1 Conclusions

In this thesis we addressed the problem of dynamic reallocation of vehicles among teams with mixed-initiative interactions. We used, as an illustration, a SEAD case study where teams of UCAVs have to strike several SAMs in a coordinated fashion. There is an off-line planning stage and an on-line execution control. The planning stage produces: a plan consisting of several tasks which have to be executed concurrently; the structure of each task which consists of a sequence of SAMS; and an initial allocation of UCAVS to teams. There is one team per task. The execution is conduced in an adversarial environment where UCAVs may be brought down by SAMs. The operator is allowed to intervene during the planning and execution control using several degrees of freedom (mixed-initiative interactions).

Assuming that the tasks are given, we developed a planning procedure that gives the minimum number of vehicles needed by each team before each attack in order to accomplish the task with a given probability of success. The plan has two objectives: it gives the initial allocation of vehicles to teams, and it provides a "baseline" to assess the performance of teams. During the execution, if a team has more vehicles than the baseline (plan), it could deliver those vehicles to teams that are below the baseline. This balancing procedure is our core problem.

In order to formalize the reallocation problem we introduce a matrix-based framework. In this framework we can easily introduce constrains to the reallocations. One such example arises when we prevent vehicles to be reallocated to another team due to the fact that

these vehicles do not have enough fuel to perform the transition and execute the reminder of the task with the target teams. More constraints could be easily added such as the number of weapons and the type of payload. We also allow the operator to constrain reallocations. This could be useful when, for example, the operator receives information that new threats are on the field.

The execution control problem is formalized as an MDP problem. In short, we want to minimize an expected cost function subject to the above constraints. The cost function has three terms: a cost due to unbalancing, a cost due to transitions, and a cost due to losing vehicles. To overcome the problem of having several (sometimes conflicting) objectives we adopt a weighted cost function where weights are tuned by the operator. The MDP problem is solved using Stochastic Dynamic Programming framework.

We provided a model for the interactions of the operator with the execution control. The model allows the operator to change some parameters of execution control. The parameters should be changed based on the experience of the operator and on emergent information which is not captured in the mathematical model. The operator is also allowed to manage the complexity of the problem. This is done by allowing the operator to trade optimality for computational speed with the help of a few guidelines. Aggregation techniques are used to speed up the computations. The variations of the value function induced by these aggregations are used as guidelines in the process of finding the optimal tradeoff between optimality and computational feasibility.

## 8.2 Future work

The complexity of the dynamic reallocation problem increases significantly with the number of vehicles, the number of teams, and the number of stages. In the future we want to assess how to handle complexity in several ways: using techniques such as approximate dynamic programming [43, 44], and understanding the behavior of the cost function when we consider sub-optimal strategies.

We also want to study what is the best control architecture for the dynamic reallocation problem. In this work we choose a hierarchical architecture but a more reactive and/or distributed architecture may be more adequate. In a reactive architecture the control

links can be established or ceased on-the-fly. Besides the control architecture, we also want to study the communication network topology, i.e. what elements of the network should be connected in order to increase the overall performance while decreasing the communication costs.

In a near future we want to take into account timing issues in the cost function and in the constraints of our dynamic reallocation problem. Basically, we want to balance the performance of teams without violating deadlines.

### 8.2.1 Other applications - surveillance

We want to address the dynamic reallocation problem in other missions rather than SEAD. The work developed in [54] and [29] for the DARPA-MICA Challenge motivated us to choose a SEAD case study. However, the models and the controllers developed in our work can be easily extended to other types of missions. Let us consider an alternative mission where the model could be applied.

Consider a large area that has to be persistently surveyed by a set of vehicles equipped with payloads such as Electro-Optic/Infra-Red systems and Syntectic Aperture Radars. Suppose that the overall area is partitioned in several small areas according to the features of the world (e.g. terrain shape) and missions requirements (e.g. scan rate - smaller areas can allow higher scan rates, i.e. a certain unit of area could be scanned more frequently). Each area is defined as a task that will be assigned to a team of vehicles. Each task is composed by stages. Each stage will be a parameterized maneuver (e.g. a follow path maneuver, a loiter maneuver to focus on points of interest, search and pursue mobile threats). Given the tasks structures, the allocation of vehicles to teams is done by some planning procedure. This procedure also returns the minimum number of vehicles needed by each team at each stage in order to fulfill the task successfully. Similar to the execution of a SEAD mission, the execution of tasks in a surveillance mission may not go as initially planned. There could be several sources of uncertainty such as weather condition (that will influence the autonomy of vehicles), failure of vehicles, and mobile threats to pursue. Thus, a dynamic reallocation of vehicles among teams could increase the overall mission's performance.

# Bibliography


[1] in *Unmanned Aircrafts Systems Roadmap 2005-2030*. USA: United States Department of Defense, 2005.

[2] P. Antsaklis and X. Koutsoukos, "Hybrid systems: review and recent projects," in *Software-Enabled Control*. Wiley-Interscience, Mar. 2003, pp. 273–298.

[3] [Online]. Available: http://www.defensetech.org/archives/002651.html

[4] M. Ownby, "Mixed initiative control of automa-teams - operational seminar," Feb 2003.

[5] R. Bencatel, J. Correia, J. Sousa, G. Gonçalves, and E. Pereira, "Video tracking control algorithms for unmanned air vehicles," in *ASME Dynamic Systems and Control Conference*, Michigan, USA, Oct. 2008.

[6] E. Pereira, R. Bencatel, J. Correia, L. Félix, G. Gonçalves, J. Morgado, and J. Sousa, "Unmanned air vehicles for coastal and environmental research," in *accepted at the 10th International Coastal Symposium (ICS 2009)*, Lisbon, Portugal, Apr. 2009.

[7] E. Pereira and J. Sousa, "Dynamic reallocation in teams of unmanned air vehicles," in *accepted at the AIAA Conference Unmanned...Unlimited*, Seattle, USA, Apr. 2009.

[8] K. Cook, "The silent force multiplier: The history and role of uavs in warfare," in *IEEE Aerospace Conference*, Big Sky, USA, Mar. 2007.

[9] JAPCC-NATO, in *The Joint Air Power Competence Centre (JAPCC) flight plan for Unmanned Aircraft Systems (UAS) in NATO*. Kalkar, Germany: NATO, Mar. 2007.

[10] in *Unmanned Systems Roadmap 2007-2032*. USA: United States Department of Defense, 2007.

[11] P. Blyenburgh, "Uav systems: Global review," in *Avionics 06 Conference*, Amsterdam, The Netherlands, Mar. 2006.

[12] C. Nehme, J. Crandall, and M. Cummings, "An operator function taxonomy for unmanned aerial vehicle missions," in *12th International Command and Control Research and Technology Symposium - Adapting C2 to the 21st Century*, Newport, USA, Jun. 2007.

[13] "Ieee standard for application and management of the systems engineering process," *IEEE Std 1220-2005 (Revision of IEEE Std 1220-1998)*, pp. 01–87, 2005.

[14] NASA, Ed., *NASA Systems Engineering Handbook*. NASA, 1995.

[15] S. M. C. D. of Defense, Ed., *System Engineering Fundamentals*. DoD, 2001.

[16] [Online]. Available: http://www.defense-update.com/features/du-3-05/feature-c4i.htm

[17] E. Olaussen and A. Karlsen, "A policy-based priority and precedence framework for military ip networks," *Military Communications Conference, 2004. MILCOM 2004. IEEE*, vol. 2, pp. 827–833 Vol. 2, Oct.-3 Nov. 2004.

[18] J. Vesecky and J. Cornwall, "Integrated design of synthetic aperture radars for unmanned aircraft," *Geoscience and Remote Sensing Symposium, 1996. IGARSS '96. 'Remote Sensing for a Sustainable Future.', International*, vol. 4, pp. 2347–2348 vol.4, May 1996.

[19] A. Muccio and T. Scruggs, "Moving target indicator (mti) applications for unmanned aerial vehicles (uavs)," in *Proceedings of the International Radar Conference 2003*, Huntsville, USA, May 2003.

[20] [Online]. Available: http://defense-update.com/features/du-2-05/feature-uav.htm

[21] J. Borky, "Payload technologies and applications for uninhabited air vehicles (uavs)," *Aerospace Conference, 1997. Proceedings., IEEE*, vol. 3, pp. 267–283 vol.3, Feb 1997.

[22] J. Egbert and R. Beard, "Low altitude road following constraints using strap-down eo cameras on miniature air vehicles," in *Proceedings of the 2007 American Control Conference*, New York, USA, Jul. 2007.

[23] E. Frew, T. McGee, Z. Kim, X. Xiao, S. Jackson, M. Morimoto, S. Rathinam, J. Padial, and R. Sengupta, "Vision-based road-following using a small autonomous aircraft," in *2004 IEEE Aerospace Conference Proceedings*, Big Sky, USA, Mar. 2004.

[24] H. Eisenbeiss, "A mini unmanned aerial vehicle (uav): System overview and image acquisition," in *International Workshop on Processing And Visualization Using High-Resolution Imagery*, Pitsanulok, Thailand, Nov. 2004.

[25] R. He, S. Prentice, and N. Roy, "Planning in information space for a quadrotor helicopter in a gps-denied environment," *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 1814–1820, May 2008.

[26] D. Casbeer, D. Kingston, R. Beard, and T. Mclain, "Cooperative forest fire surveillance using a team of small unmanned air vehicles."

[27] A. Ryan, X. Xiao, S. Rathinam, J. Tisdale, M. Zennaro, D. Caveney, R. Sengupta, and J. Hedrick, "A modular software infrastructure for distributed control of collaborating uavs," in *Proceedings of the AIAA Conference on Guidance, Navigation, and Control*, Keystone, USA, Aug. 2006.

[28] A. Ryan, J. Tisdale, M. Godwin, D. Coatta, D. Nguyen, S. Spry, R. Sengupta, and K. Hedrick, "Decentralized control of unmanned aerial vehicle sensing missions," in *Proceedings of the American Controls Conference*, New York, USA, Jul. 2007.

[29] P. Varaiya, in *Hierarchical control of semi-autonomous teams under uncertainty (HICST) - Final report of Darpa Contract F33615-01-C-3150.* USA: DARPA, may 2004.

[30] [Online]. Available: http://www.uavm.com/uavregulatory/airworthinesscertification.html

[31] E. Lee and P. Varaya, *Structure and interpretation of signals and systems.* Addison-Wesley, February 2003.

[32] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems.* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[33] C. Tomlin, "Lecure notes on hybrid systems: Modeling, analysis, and control," March 1995.

[34] C. Tomlin, S. Boyd, I. Mitchell, A. Bayen, M. Johansson, and L. Xiao, "Computational tools for the verification of hybrid systems," in *Software-Enabled Control*. Wiley-Interscience, Mar. 2003, pp. 369–392.

[35] O. Maler, "What is verification?" August 2008. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/477.html

[36] A. A. Kurzhanskiy and P. Varaiya, "Ellipsoidal techniques for reachability analysis of discrete-time linear systems," *Automatic Control, IEEE Transactions on*, vol. 52, no. 1, pp. 26–38, Jan. 2007.

[37] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

[38] R. E. Bellman, *Dynamic Programming*. Dover Publications, Incorporated, 2003.

[39] D. Bertzekas, "Lecture slides on dynamic programming," 2003.

[40] A. Nilim and L. E. Ghaoui, "Robust markov decision processes with uncertain transition matrices," University of California at Berkeley, Tech. Rep., 2004.

[41] F. P. S. David S. Alberts, John J. Garstka, *Network centric warfare : developing and leveraging information superiority 2nd Edition (Revised)*. CCRP publication series, August 1999.

[42] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, pp. 49–95, 1996.

[43] D. P. de Farias and B. V. Roy, "The linear programming approach to approximate dynamic programming," *Oper. Res.*, vol. 51, no. 6, pp. 850–865, 2003.

[44] D. P. D. Farias and B. V. Roy, "On constraint sampling for the linear programming approach to approximate dynamic programming," *Mathematics of Operations Research*, vol. 29, p. 2004, 2001.

[45] R. M. Murray, Ed., *Control in an Information Rich World: Report of the Panel on Future Directions in Control, Dynamics, and Systems*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2003.

[46] R. M. Murray, J. Hauser, A. Jadbabaie, M. B. Milam, N. Petit, W. B. Dunbar, and R. Franz, "Semidefinite programming relaxations of non-convex problems in control and combinatorial optimization," in *In Software-Enabled Control: Information Technology for Dynamical Systems*. Wiley-Interscience, 2002, pp. 149–174.

[47] A. Nilim and L. E. Ghaoui, "Robustness in markov decision problems with uncertain transition matrices," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004.

[48] S. Boyd and L. Vandenberghe, "Online control customization via optimization-based control," in *Communications, Computation, Control and Signal Processing: A Tribute to Thomas Kailath, A. Paulraj, V. Roychowdhuri, and C. Schaper*. Kluwer, 1997, pp. 279–288.

[49] R. A. Freeman and P. V. Kototovic, *Robust nonlinear control design: state-space and Lyapunov techniques*. Cambridge, MA, USA: Birkhauser Boston Inc., 1996.

[50] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*, Jun. 2008, manuscript preprint. Electronically available at http://coordinationbook.info.

[51] J. Lygeros, D. N. Godbole, and S. Sastry, "Verified Hybrid Controllers for Automated Vehicles," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 522–539, 1998. [Online]. Available: http://control.ee.ethz.ch/index.cgi?action=details;id=2549;page=publications

[52] A. Deshpande and P. Varaiya, "Viable control of hybrid systems," in *Hybrid Systems*, 1994, pp. 128–147.

[53] A. Deshpande, A. Gollu, and L. Semenzato, "The shift programming language and run-time system for dynamic networks of hybrid automata," University of California at Berkeley, Tech. Rep., 1997.

[54] J. Sousa, T. Simsek, and P. Varaiya, "Task planning and execution for uav teams," *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 4, pp. 3804–3810 Vol.4, Dec. 2004.

[55] S. F. El-Zoghdy, "Certain performance aspects of optimal load balancing in distributed computer systems," Ph.D. Dissertation, University of Tsukuba, March 2003.

[56] G. Tel, *Introduction to distributed algorithms*, ch. Introduction: Distributed Systems, p. 14.

[57] J. Li and H. Kameda, "A decomposition algorithm for optimal static load balancing in tree hierarchy network configurations," *IEEE Trans. Parallel and Distributed Systems*, vol. 5, pp. 540–548, 1994.

[58] ——, "Load balancing problems for multiclass jobs in distributed/ parallel computer systems," *IEEE Trans. Computer*, vol. 47, pp. 322–332, 1998.

[59] S. Sastry, D. Godbole, J. Malik, R. Sengupta, and O. Shakernia, "Interim progress report: Intelligent control architectures for unmanned air vehicles," dec 1997.

[60] G. Pappas, G. Lafferriere, and S. Sastry, "Hierarchically consistent control systems," *Automatic Control, IEEE Transactions on*, vol. 45, no. 6, pp. 1144–1160, Jun 2000.

[61] W. Ren and R. W. Beard, *Distributed Consensus in Multi-Vehicle Cooperative Control*. Springer Verlag.

[62] J. Allen, C. Guinn, and E. Horvtz, "Mixed-initiative interaction," *IEEE Intelligent Systems*, vol. 14, no. 5, pp. 14–23, 1999.

[63] C. E. Nehme, "Modeling human supervisory control in heterogeneous unmanned vehicle systems," Ph.D. Dissertation, Massachusetts Institute of Technology, Feb 2009.

[64] "Suppression of enemy air defenses (sead)," *United States Marine Corps Warfighters Publications*, vol. 3-22.2, 2001.

[65] J. Jelinek and D. Godbole, "Model predictive control of military operations," *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, vol. 3, pp. 2562–2567 vol.3, 2000.

[66] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, February 1993.

[67] R. Caballero, G. M. GómezT, L. Rey, and F. Ruiz, "Goal programming with dynamic goals," *Journal of Multi-Criteria Decision Analysis*, no. 7, pp. 217–229, 1998.

[68] K. Chatterjee, "Markov decision processes with multiple long-run average objectives," in *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science*, December 2007, pp. 473–484. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/466.html

[69] M. Mendel and T. Sheridan, "Filtering information from human experts," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 19, no. 1, pp. 6–16, Jan/Feb 1989.

[70] C. Nehme, B. Mekdeci, J. Crandall, and M. L. Cummings, "The impact of heterogeneity on operator performance in future unmanned vehicle systems," *The International C2 Journal - Special Issue: Representing Human Decision Making in Constructive Simulations for Analysis*, vol. 2, no. 2, nov 2008.