UNIVERSITY OF CALIFORNIA

Los Angeles

Linear-Programming-Based Multi-Vehicle Path Planning with Adversaries

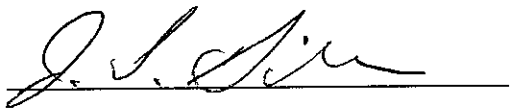
A thesis submitted in partial satisfaction
of the requirements for the degree Master of Science
in Mechanical Engineering

by

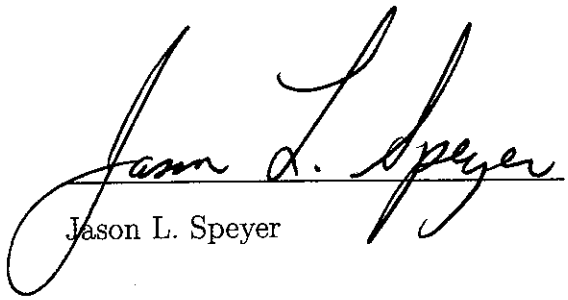
Georgios Christos Chasparis

2004

The thesis of Georgios Christos Chasparis is approved.



James S. Gibson



Jason L. Speyer

Jeff S. Shamma, Committee Chair

University of California, Los Angeles

2004

To my parents

TABLE OF CONTENTS

List of Figures . . . . . **viii**

List of Symbols . . . . . **xi**

Acknowledgments . . . . . **xix**

Abstract of the Thesis . . . . . **xx**

1 Introduction **1**

1.1 Motivation 1

1.2 Applications 3

1.3 Multi-vehicle path planning 4

1.3.1 Coordination variables 5

1.3.2 Voronoi-based polygonal paths 6

1.3.3 Model predictive control 8

1.3.4 Mixed-integer linear programming (MILP) 9

1.3.5 Dynamic programming 10

1.3.6 Remarks 12

1.4 Objective of the thesis 12

1.5 Thesis outline 13

2 Resource Allocation Model **15**

2.1 Introduction 15

2.2 State-space representation 15

2.2.1 Selection of the states . . . 15
2.2.2 State evolution . . . 17
2.2.3 State and control constraints . . . 20
2.3 System specifications . . . 22
2.4 Remarks . . . 24

3 Adversarial Environment . . . 25
3.1 Introduction . . . 25
3.2 State-space representation . . . 26
3.3 Problem reformulation . . . 28
3.3.1 Model simplifications . . . 29
3.3.2 Creation of the enemy's feedback matrix . . . 31
3.4 Optimization set-up . . . 33
3.4.1 Objective function . . . 33
3.4.2 Dynamic constraints . . . 34
3.4.3 State constraints . . . 35
3.4.4 Control constraints . . . 36
3.4.5 Convex optimization problem . . . 38
3.4.6 Equivalent linear optimization problem . . . 39
3.4.7 Alternative objective function . . . 42
3.4.8 Feasibility of the optimization problem . . . 44
3.5 Receding horizon philosophy . . . 44
3.6 Remarks . . . 47

4 Multi-Vehicle Path Planning . . . 48

4.1 Introduction 48
4.2 The RoboFlag competition 48
4.3 Linear-programming-based defense planning 50
4.3.1 Binary constraints 51
4.3.2 Defense-zone constraints 53
4.3.3 Mixed integer programming optimization problem 53
4.3.4 Linear programming relaxation 56
4.3.5 Computation of a suboptimal solution 57
4.3.6 Control algorithm for defense 59
4.3.7 Computational simplifications 60
4.3.8 Alternative objective functions 61
4.3.9 Remarks 64
4.4 Linear-programming-based attack planning 65
4.4.1 Linear programming relaxation 65
4.4.2 Computation of a suboptimal solution 66
4.4.3 Control algorithm for attack 69
4.5 The enemy's feedback matrix 70
4.5.1 Stochastic feedback matrix for defense 71
4.5.2 Alternative stochastic feedback matrix for defense 73
4.5.3 Stochastic feedback matrix for attack 74
4.5.4 Velocity-based stochastic feedback matrix 76
4.5.5 Combined stochastic feedback matrix 79
4.6 Remarks 79

5 Simulations **83**

5.1 Introduction 83

5.2 The "Cornell RoboFlag Simulator" 83

5.3 A RoboFlag simulator created in Matlab 86

5.4 Simulations and Discussion 86

5.5 Remarks 89

6 Conclusions and Future Work **96**

References **98**

LIST OF FIGURES

2.1 Arena of two sectors. 16

2.2 Block diagram of resources flow. 22

3.1 Block diagram of friendly and enemy resources flow with attrition. . . 28

3.2 Block diagram of the simplified model for friendly and enemy resources flow. 30

3.3 Basic structure of Receding Horizon Philosophy. 46

4.1 A simplified version of the RoboFlag competition, *Roboflag Drill.* . . 49

4.2 The playing area is separated into sectors. 52

4.3 The two polyhedra P_{LP} and P_{MI} contain exactly the same set of integer solutions. 56

4.4 A "low-energy" zone is defined about the defense zone. 63

4.5 Possible directions of motion in an arena of sectors. 77

4.6 Probability distribution of three attackers after $N_p = 2$ stages based on the stochastic feedback matrix for defense. 80

4.7 Probability distribution of three attackers after $N_p = 2$ stages based on the alternative stochastic feedback matrix for defense with $\bar{n}^a_{dsn} = 6$. 81

4.8 Probability distribution of two defenders after $N_p = 2$ stages based on the stochastic feedback matrix for attack. 82

5.1 A general architecture of the "Cornell RoboFlag Simulator." 85

5.2 A control algorithm for defense with $N_p = 6$ is applied in the RoboFlag simulator created in Matlab. A stochastic feedback matrix for defense with $\bar{n}^a_{dsn} = 6$ is used. The attackers follow pre-specified paths unknown to the defenders. 90

5.3 A control algorithm for defense with $N_p = 6$ is applied in the "Cornell RoboFlag Simulator". A stochastic feedback matrix for defense with $\bar{n}^a_{dsn} = 6$ is used. The attackers follow pre-specified paths unknown to the defenders. 91

5.4 A control algorithm for defense with $N_p = 6$ is applied in the RoboFlag simulator created in Matlab. A stochastic feedback matrix for defense with $\bar{n}^a_{dsn} = 6$ combined with a velocity-based feedback matrix is used. The attackers follow pre-specified paths unknown to the defenders. . . 92

5.5 A control algorithm for defense with $N_p = 6$ is applied in the RoboFlag simulator created in Matlab. The defenders implement a stochastic feedback matrix for defense with $\bar{n}^a_{dsn} = 6$. The attackers use a control algorithm for attack with $N_p = 6$, $w^a_{dz} = 0.01$ and $w^a_d = 0.99$ in combination with a stochastic feedback matrix for attack. 93

5.6 A control algorithm for defense with $N_p = 6$ is applied in the RoboFlag simulator created in Matlab. The defenders implement a stochastic feedback matrix for defense with $\bar{n}^a_{dsn} = 6$. The attackers use a control algorithm for attack with $w^a_{dz} = 0.01$ and $w^a_d = 0.99$ only at the first and last stage of the optimization horizon, otherwise $w^a_{dz} = 0$ and $w^a_d = 1$. The attackers also implement a stochastic feedback matrix for attack. 94

5.7 A control algorithm for defense with $N_p = 6$ is applied in the RoboFlag simulator created in Matlab. The defenders use the objective function of Section 4.3.8 with weights $w_a^d = 0.95$ and $w_{lz}^d = 0.05$ and a stochastic feedback matrix for defense with $\bar{n}_{dsn}^a = 6$. The attackers use a control algorithm for attack with $w_{dz}^a = 0.01$ and $w_d^a = 0.99$ only at the first and last stage of the optimization horizon, otherwise $w_{dz}^a = 0$ and $w_d^a = 1$. The attackers also use a stochastic feedback matrix for attack. 95

LIST OF SYMBOLS

a	abbreviation of "attackers"
$\mathbf{b}_{s_i,r_j}$	the vector whose inner product with $\mathbf{u}$ is the change in the level of resource type r_j at sector s_i
$\mathbf{b}_{s_i,r_j,\text{in}}$	the vector whose inner product with $\mathbf{u}$ is the level of resource type r_j that enters sector s_i
$\mathbf{b}_{s_i,r_j,\text{out}}$	the vector whose inner product with $\mathbf{u}$ is the level of resource type r_j that exits from sector s_i
$\beta(s_i, s_k)$	minimum distance (in sectors) from sector s_i to sector s_k
d	abbreviation of "defenders"
dz	abbreviation of "defense zone"
e	abbreviation of "enemy"
f	abbreviation of "friendly"
$g^a_{s_i,r_j,t}$	probability that the attackers' resource type $r_j \in \mathcal{R}^a$ is at sector $s_i \in \mathcal{S}$ after t time intervals
$g^e_{s_k \leftarrow s_i,r_j}$	position-based probability that the enemy resource type $r_j \in \mathcal{R}^e$, which lies in sector $s_i \in \mathcal{S}$, will move to sector $s_k \in \{\mathcal{SN}^e(s_i, r_j) \cup s_i\}$
$g^a_{s_k \leftarrow s_i,r_j}$	position-based probability that the attackers' resource type $r_j \in \mathcal{R}^a$, which lies in sector $s_i \in \mathcal{S}$, will move to sector $s_k \in \{\mathcal{SN}^a(s_i, r_j) \cup s_i\}$

$g^{v}_{s_k \leftarrow s_i, r_j}$	velocity-based probability that the enemy resource type $r_j \in \mathcal{R}^e$, which lies in sector $s_i \in \mathcal{S}$, will move to sector $s_k \in \{\mathcal{SN}^e(s_i, r_j) \cup s_i\}$
$g^{a}_{s_k \leftarrow s_i, r_j, \text{alt}}$	position-based probability that the attackers' resource type $r_j \in \mathcal{R}^a$, which lies in sector $s_i \in \mathcal{S}$, will move to sector $s_k \in \mathcal{DSN}^a(s_i, r_j, \bar{n}^a_{dsn})$
lz	abbreviation of "low-energy zone"
$\mathbf{l}$	the vector of attrition functions
l_{s_i, r_j}	the attrition function of resource type r_j at sector s_i
n_r	number of resources
n_s	number of sectors in the arena
n_x	number of states
$\bar{n}^a_{sn}(s_i, r_j)$	number of neighboring sectors of sector s_i including itself that can be reached by the attackers' resource type r_j
$\bar{n}^a_{dsn}(s_i, r_j)$	number of neighboring sectors of sector s_i including itself that can be reached by the attackers' resource type (unit) r_j and are the most probable next positions for that unit based on their distance from the defense zone
r_j	resource type
s_i	sector number
s_{dz}	sector number that corresponds to the center of the defense zone

$\mathbf{s}^a_{mp,t}$	the vector of the most probable positions (sectors) of the attackers after t time intervals
$\mathbf{u}$	control vector
$\mathbf{u}[t]$	control vector at time t
$\mathbf{u}^*[t]$	optimal control vector at time t
$\tilde{\mathbf{u}}[t]$	suboptimal control vector at time t
$\mathbf{u}_{r_j}$	control vector that is composed of the transitions of resource type r_j in the arena of sectors
$\mathbf{u}_{s_i,r_j}$	control vector that contains only the levels of resource type r_j that enters sector s_i
$u_{s_i \leftarrow s_k,r_j}$	the level of resource type r_j that is transferred from sector s_k to sector s_i
$\mathbf{v}_{s_i,r_j}$	current direction of motion of resource type $r_j \in \mathcal{R}^e$ that lies in sector $s_i \in \mathcal{S}$
$\mathbf{v}^+_{s_k,r_j}$	next direction of motion of resource type $r_j \in \mathcal{R}^e$ that moves from s_i to $s_k \in \{\mathcal{SN}^e(s_i, r_j) \cup s_i\}$
w^a_{dz}	the weight that the attackers attach to getting closer to the defense zone
w^a_d	the weight that the attackers attach to staying away from the defenders
w^d_a	the weight that the defenders attach to getting closer to the attackers

w_{lz}^{d}	weight that the defenders attach to getting closer to the low-energy zone
$\mathbf{x}$	state vector
$\mathbf{x}_1$	state vector of one resource type
$\mathbf{x}^{+}$	state vector at the next stage
$\mathbf{x}[t]$	state vector at time t
$\mathbf{x}_{r_j}$	state vector of resource type r_j
$\mathbf{x}_{max}$	state vector of the maximum resource levels
x_{s_i,r_j}	the level of resource type r_j at sector s_i
x_{s_i,r_j}^{+}	the level of resource type r_j at sector s_i at the next stage
$x_{s_i,r_j,\max}$	positive upper bound of the level of resource type r_j at sector s_i
$\mathbf{y}$	output vector of a multi-output system
z	z-transform variable
$\mathbf{B}$	the matrix whose product with $\mathbf{u}$ is the change in the state vector at the next stage
$\mathbf{B}_{\text{in}}$	the matrix whose product with $\mathbf{u}$ is the vector of resource levels that enter each sector
$\mathbf{B}_{\text{out}}$	the matrix whose product with $\mathbf{u}$ is the vector of resource levels that exit from each sector

$\mathcal{C}(\mathbf{x})$	the set of control vectors that satisfy the continuity constraints given the current state vector $\mathbf{x}$
$\mathcal{DSN}^a(s_i, r_j, \bar{n}^a_{dsn})$	the set of $\bar{n}^a_{dsn}$ neighboring sectors of sector $s_i \in \mathcal{S}$ including itself that can be reached by the attackers' resource type (unit) $r_j \in \mathcal{R}^a$ and are the most probable next positions for that unit based on their distance from the defense zone
$\mathbf{G}^e$	stochastic feedback matrix of the enemy resources
$\mathbf{G}^e_{r_j}$	stochastic feedback matrix of the enemy resource type $r_j \in \mathcal{R}^e$
$\mathbf{G}^a$	stochastic feedback matrix of the attackers' resources
$\mathbf{G}^a_{r_j}$	stochastic feedback matrix of the attackers' resource type $r_j \in \mathcal{R}^a$
$\mathbf{G}^a_{r_j,\text{alt}}$	alternative stochastic feedback matrix of the attackers' resource type $r_j \in \mathcal{R}^a$
$\mathbf{G}^v$	velocity-based stochastic feedback matrix of the enemy resources
$\hat{\mathbf{G}}(z)$	transfer function of a discrete-time system
$\mathcal{N}$	the set of natural numbers
N_p	optimization horizon
$\mathcal{P}$	the set of state vectors that satisfy the positivity constraints

$\mathcal{R}$	the set of resource types
$\Re$	the set of real numbers
$\Re_+$	the set of non-negative real numbers
$\Re^n$	the set of real vectors with dimension n
$\Re^n_+$	the set of non-negative real vectors with dimension n
$\mathcal{S}$	the set of sectors in the arena
$\mathcal{S}^{dz}$	the set of sectors in the defense zone
$\mathcal{S}^{lz}$	the set of sectors in the low-energy zone
$\mathcal{SN}(s_i)$	the set of neighboring sectors of sector s_i
$\mathcal{SN}(s_i, r_j)$	the set of neighboring sectors of sector s_i that can be reached by resource type r_j
$\mathbf{T}_c$	block diagonal matrix whose diagonal entries are equal to $\mathbf{B}_{\text{out}}$
$\mathbf{T}_{xu}$	the matrix whose product with the control vector from stage 0 to stage $(N_p - 1)$, $U_{0\to(N_p-1)}$, is the state vector from stage 1 to stage N_p, $X_{1\to N_p}$, for assumed zero initial state
$\mathbf{T}_{xu,c}$	the matrix whose product with the control vector from stage 0 to stage $(N_p - 1)$, $U_{0\to(N_p-1)}$, is the state vector from stage 0 to stage $N_p - 1$, $X_{0\to(N_p-1)}$, for assumed zero initial state

$\mathbf{T}_{xx_0}$	the matrix whose product with the initial state vector, $\mathbf{x}[0] = \mathbf{x}_0$, is the state vector from stage 1 to stage N_p, $X_{1\to N_p}$, for assumed zero control input
$\mathbf{T}_{xx_0,c}$	the matrix whose product with the initial state vector, $\mathbf{x}[0] = \mathbf{x}_0$, is the state vector from stage 0 to stage (N_p-1), $X_{0\to(N_p-1)}$, for assumed zero control input
$\mathbf{T}_{xx_0,G}$	the matrix whose product with the initial state vector $\mathbf{x}[0] = \mathbf{x}_0$ and for assumed state feedback matrix G is the state vector from stage 1 to stage N_p, $X_{1\to N_p}$
$\mathbf{T}_{xx,\mathrm{id}}$	block row matrix whose entries are equal to the identity matrix
$\mathbf{U}$	control vector for the whole optimization horizon
$\mathbf{U}^*$	optimal control vector for the whole optimization horizon
$\mathbf{U}_{t_1\to t_2}$	control vector from time t_1 to time t_2
$\mathcal{U}_{\mathrm{in}}$	the function that maps a triple of the form (s_i, r_j, s_k), where $s_i \in \mathcal{S}$, $r_j \in \mathcal{R}$ and $s_k \in \mathcal{SN}(s_i, r_j)$, to the row number of $\mathbf{u}$ that corresponds to $u_{s_i\leftarrow s_k, r_j}$
$\mathcal{U}_{\mathrm{out}}$	the function that maps a triple of the form (s_k, r_j, s_i), where $s_i \in \mathcal{S}$, $r_j \in \mathcal{R}$ and $s_k \in \mathcal{SN}(s_i, r_j)$, to the row number of $\mathbf{u}$ that corresponds to $u_{s_k\leftarrow s_i, r_j}$
$\mathcal{UN}(s_i, r_j)$	the set of row numbers of $\mathbf{u}$ which correspond to the control inputs $\left\{u_{s_k\leftarrow s_i, r_j} \mid s_k \in \mathcal{SN}(s_i, r_j)\right\}$

$\mathbf{X}$	the vector of resource levels for the whole optimization horizon
$\mathbf{X}_1$	the vector of resource levels of one resource type for the whole optimization horizon
$\mathbf{X}_{\max}$	the vector of maximum resource levels for the whole optimization horizon
$\mathbf{X}_{t_1 \to t_2}$	state vector from time t_1 to time t_2
$\mathcal{X}(\mathbf{x})$	the set of reachable state vectors given current state vector $\mathbf{x}$
$\mathbf{Y}$	the vector of slack variables
$\mathcal{Z}$	the set of integer numbers
$\mathcal{Z}_+$	the set of non-negative integer numbers
$\mathcal{Z}^n$	the set of integer vectors with dimension n
$\mathcal{Z}_+^n$	the set of non-negative integer vectors with dimension n

ACKNOWLEDGMENTS

I would like to express my sincere gratefulness to my advisor, Professor Jeff S. Shamma, for his guidance and encouragement during the completion of my thesis.

I would like also to thank my colleagues in the Control Lab of the Mechanical Engineering Department at UCLA for our discussions and their valuable advices.

Thesis supported by AFOSR/MURI grant #F49620-01-1-0361.

ABSTRACT OF THE THESIS

Linear-Programming-Based
Multi-Vehicle Path Planning with Adversaries

by

Georgios Christos Chasparis

Master of Science in Mechanical Engineering
University of California, Los Angeles, 2004
Professor Jeff S. Shamma, Chair

Coordination is of vital importance in multi-vehicle systems and is always a challenge for control engineers. In this thesis, a linear-programming-based path planning algorithm is developed for deriving optimal paths for a group of autonomous vehicles in an adversarial environment, such as the RoboFlag competition. In this method, both friendly and enemy vehicles are modelled as different resource types in an arena of sectors. Simple model simplifications are introduced to allow the use of linear programming in conjunction with a receding horizon implementation for vehicle path planning. Since the enemy's future moves are unknown, various stochastic models based on their current position and/or velocity are used to describe their possible future trajectories. Several objective functions are examined against various enemy's feedback laws, while their utility is tested in the "Cornell RoboFlag Simulator." Results show that a linear-programming-based planning in combination with a simple enemy's model can be used for effective multi-vehicle path planning in an adversarial environment.

CHAPTER 1

Introduction

1.1 Motivation

The main features of autonomous control systems, such as robotic systems, are determined by the need to accept high level descriptions of tasks and execute them without any human intervention. These high level descriptions of commands determine the goal of the autonomous control system rather than the way in which this goal will be accomplished.

More specifically, repeatability of actions is not the main purpose of today's robotic systems. Early robot designs were primarily used in applications, such as manufacturing, where their actions were always pre-specified, while most of the times their tasks did not include any unexpected events. In such applications, robots must follow specific paths, while an inner feedback control loop usually guarantees small tracking errors.

However, in modern robotics applications like *navigation* or *obstacle avoidance* a higher level of control objectives is necessary. For example, in an obstacle avoidance problem a *high level* control objective would be the real-time derivation of a "safe" path, so that robots avoid all the obstacles in the environment. On the other hand, minimization of the tracking error is considered as a *low level* control objective. In other words, *hierarchy* of the control objectives in modern robotics applications is important.

The distinction between the different levels of control objectives is usually deter-

mined by the different types of *uncertainty* which the controller needs to accommodate. For example, uncertainties like parametric plant uncertainty, unmodelled plant dynamics, disturbance signals and sensor noise are some of the most important handicaps in achieving optimal response in a tracking task. Such types of uncertainty can be characterized as a *low uncertainty level*.

On the other hand, in problems like *navigation* in domestic environments, environmental conditions change all the time and as a result *the uncertainty level is higher*. A more complicated problem could involve a group of adversaries that try to prevent robots from completing their task. In that case, robots must be able not only to complete their task, but also to avoid being harmed from the adversaries. In other words, in such environments the planning problem becomes more difficult and a higher control level is necessary.

Although the problem of eliminating the effect of low uncertainty level types has extensively been studied, the problem of a higher control level, such as multi-vehicle coordination, in the face of higher uncertainty levels, such as continuous change of the environmental conditions, is still an open issue. In particular, although this problem can be formulated as an optimization problem, inclusion of the high uncertainty level and real time derivation of the optimal solution are some of the most important challenges in multi-vehicle coordination.

In such applications, it may also be preferable to use a large number of robots. For example, due to unexpected events several damage can be caused to the control devices. Having several robots working on the same task maximizes the chances that the project will be completed properly, since the loss of some robots does not affect the rest of them. Although this system is more flexible due to the "strength in numbers", more complicated control laws must be used. The robots must be able to complete their task without any human intervention and independently of the possible uncertainties of the environment.

In conclusion, the most important features that contribute to the challenge of *multi-vehicle control systems* are

- hierarchy of control objectives and distinction between high and low control level
- formulating an optimization problem that guarantees multi-vehicle coordination
- dealing with uncertainty and its inclusion to the optimization problem
- dealing with the complexity of the optimization problem when the number of vehicles increases

1.2 Applications

Autonomous robots have already been used to perform missions in hazardous environments, such as exploration of Mars and operations in nuclear power plants. One of the these applications is the development of more intelligent Unmanned Aerial Vehicles (UAVs). The main objective is to arrive at the given target within the prespecified time, while maximizing the safety of the UAVs. Future UAVs will be designed to make tactical decisions autonomously and will be integrated into teams that coordinate to achieve high-level goals.

Moreover, a significant challenge in the high level control logic is the RoboFlag compeptition, which is described by D'Andrea and Murray in [12]. Two teams of robots compete to each other in a game where each team's objective is both to capture the opponent's flag and to thwart opponents from capturing its own flag. Due to a limited communications rate and a limited information sensing, this game provides a test-bed for high level control of multivehicle systems.

Both UAVs' coordination and RoboFlag competition addresses some of the most important issues of multi-vehicle control systems, such as hierarchy of control objec-

tives. More specifically, a quite simple architecture for cooperative control consists of two control levels, the vehicle control level and the team control level. In the vehicle control level (low control level) the vehicle tracks the desired trajectory imposed by the team control level (high control level). Several other architectures can be used depending on the complexity of the system, such as the one described by Chandler in [9].

Furthermore, in both applications, uncertainty of the environment plays an important role in designing optimal paths, while a large number of vehicles or a complex information structure increases the complexity of the path planning. Therefore, the choice of an efficient path planning scheme, which usually is a complex optimization problem, is vital for the efficiency of the autonomous control system.

1.3 Multi-vehicle path planning

The uncertainty of the environment and the complexity of a multi-vehicle control system creates several problems in the path planning scheme, or, in other words, in the computation of the optimal trajectories of the vehicles. In particular, such an optimization problem must provide

- a team-optimal solution
- on-line inclusion of all possible uncertainties of the environment
- computational efficiency

To this end, most of the recent work on multi-vehicle path planning focuses on seeking a method that will provide the above three characteristics. However, usually these type of optimization problems is more difficult to solve than Lagrange variational problems. For this reason, researchers have resorted to several simplifications, such as

discretizing of time and open-loop control. On the other hand, these simplifications do not exclude the case that the attendant optimization problem will not be large-scale and untractable.

Therefore the derivation of an efficient multi-vehicle path planning is one of the greatest challenges for the control engineers. Some of the optimization methods that have been used for multi-vehicle path planning are based on

- Coordination variables
- Voronoi-based polygonal paths
- Model predictive control
- Mixed-integer linear programming
- Dynamic programming

1.3.1 Coordination variables

This method of cooperative control is primarily based on the notion of *coordination variable* and *coordination function*. A general approach is introduced by McLain and Beard in [27]. According to this approach, the information that must be shared to facilitate cooperation is collected into a single vector quantity called the *coordination variable*.

The second main ingredient of this cooperative control strategy is the notion of a *coordination function*. The idea is to quantify how changing the coordination variable impacts the individual myopic objectives, and then to use this information to modify the coordination variable (e.g. the coordination function describes the cost to an individual UAV of achieving different values of the coordination variable that are feasible for the UAV).

McLain and Beard, [27], also provide a formal mechanism for introducing the type of team feedback that coordination function allow. Based on the team optimal coordination variable, UAV trajectories are determined in a decentralized fashion on the individual UAVs. This decomposition of the cooperative path planning results in significant simplification of team-level planning process and a substantial reduction in the volume of information communicated among UAVs.

Several simulations are shown in [27], where UAVs' objective is to avoid several threats while meeting the timing constraints imposed by their mission. Although this approach introduces an efficient means for formulation and solution of team-optimal cooperation problems, the location of the considered threats are deterministically known, i.e., uncertainties of the environment are not taken into account.

1.3.2 Voronoi-based polygonal paths

The majority of the papers on mission planning of UAVs construct a Voronoi-based polygonal path from the currently known locations of the threats to generate an initial path. A feasible flight trajectory is ensured by joining the Voronoi edges with arcs, where the radius is determined by a pre-specified vehicle turning radius. Cooperative classification is the task of optimally and jointly using multiple vehicles' sightings to maximize the probability of correct target classification.

Taken in different combinations, the edges of the Voronoi diagram provides a set of paths from the starting point to the target. Among those, Chandler and Pachter in [7] choose the lowest-cost flyable path, while Chandler in [9] chooses the path that minimizes exposure to radar.

Then the initial path should be refined to make sure that it is within the dynamic constraints of the UAV. Several different methods are used for this purpose, such as discretizing and smoothing by McLain and Chandler in [26], and virtual mass-virtual force approach by Bortoff in [5]. It is also assumed that *the location of the threats*

and their effects are deterministically known, even in the case of pop-up threats, so the effects of uncertainties in the locations of the radar sites are not considered.

Instead of deterministically specifying the locations of the threats, Dogan in [14] introduces a probabilistic approach that can be applied to several stages of mission planning of UAVs. According to this approach, a probability of threat due to multiple sources at a given position is assumed to be known and the chosen path minimizes the probability of getting disabled for a UAV to fly to a target through an area of threats.

Although the problem can be set up as a variational calculus problem, due to technical difficulties, Dogan develops a flight strategy that is parameterized to change the weighting of finding a shorter path or finding a path that will give smaller probability of getting disabled. Moreover, the paths are generated without discretizing the area of operation. However, there are several problems that arise by following this approach. In particular, global strategies might be computationally inefficient, while simulation experiments showed that the path generated by the strategy might get in a limit cycle close to the target position, but not close enough to attain it.

Moreover, Jun and D'Andrea in [21] propose a path planning algorithm based on a map of the probability of threats (adversarial environment), which can be built from a priori surveillance data. A path planning method for UAVs is proposed using a probability map. The approach in this method is similar to those in [5] and [8] in that they decompose the problem into two steps - first the generation of a preliminary polygonal path by using a graph, and then a refinement of the path. They also consider the effects of moving threats, changes in the probability map, and perform an analysis of the effects of refinement on the initial path.

1.3.3 Model predictive control

Several classes of multi-vehicle systems can be modelled as hybrid systems, namely hierarchical systems constituted by dynamical components and logical/discrete components. Especially in the case where dynamical and logical facts are dramatically interdependent, one of the suggested approaches to designing feedback controllers is based on *model predictive control* described by Bemporad and Morari in [3].

More specifically, in [3] a predictive control scheme is proposed which is able to stabilize systems described by linear dynamic equations subject to linear mixed-integer inequalities, i.e., inequalities involving both continuous and binary variables. Model predictive control is based on the *receding horizon philosophy* according to which a sequence of future control actions is chosen according to an optimization problem which is based to a prediction of the future evolution of the system. However, this sequence of future control actions is applied until a new measurement of the output of the system is available.

It is preferable to use this method when the prediction of the future evolution of the system cannot be accurate due to several reasons, such as unknown future disturbances. In this case, it is desirable to be able to rerun the optimization and uplink the new solution, thus achieving the benefit of feedback action. Due to the presence of integer variables, the optimization could be a *mixed integer quadratic programming* (MIQP) problem, [3]. For some classes of multi-vehicle systems this optimization procedure is a *mixed integer linear programming* (MILP) problem, which has been extensively used for multi-vehicle control tasks and is examined in the following section.

Finally, model predictive control has also been used in multi-vehicle formations, [15]. However, the problem of multi-vehicle formations will not be examined in this thesis, since we restrict our attention to multi-vehicle path planning with adversaries.

1.3.4 Mixed-integer linear programming (MILP)

According to this procedure, the system is composed of continuous and discrete states (a hybrid system) with linear dynamics subject to inequality constraints and logical rules. In this formulation robot dynamics are modelled with second order differential equations. The resulting optimization problem is converted to a MILP.

Two methods are compared by Richards and Bellingham in [31] for solving the optimization that combines task assignment, subjected to UAV capability constraints, and path planning, subjected to dynamics, avoidance and timing constraints. The first method expresses the entire problem as a single MILP. This method is guaranteed to find the globally optimal solution to the problem, but it is computationally intensive.

The second method employs an approximation for rapid computation of the cost of many different trajectories. This enables the assignment and trajectory problems to be decoupled and partially distributed, offering much faster computation. The approximate method offers much faster solution times, but could yield suboptimal results. The final problem is again a MILP problem. With six vehicles, twelve waypoints, and numerous no-fly-zones (obstacles), this problem is too large to be solved by the first method in any practical computation time. However, the approximate method found the solution in 27 seconds.

The utility of MILP in a simplified version of the RoboFlag competition, the "RoboFlag drill", is examined by Earl and D'Andrea in [16] and [17]. The strategy is implemented with a centralized controller with perfect knowledge of the system, perfect access to all states, and with the ability to transmit control signals to the robots instantaneously. The objective is to compute a set of control inputs that minimizes the number of opponents that enter our team's protected zone over the duration of the game. Since the opponents move along a straight line with constant

velocity, their motion and future positions are deterministically known. Moreover, the procedure is limited computationally, since the problem does not scale well with increased number of agents.

In [34], Schouwenaars presents an approach to fuel-optimal path planning of multiple vehicles using a combination of linear and integer programming. The problem is to find the optimal path between two states for a single vehicle or a group of vehicles. An approximate model of aircraft dynamics using only linear constraints is developed, enabling the MILP approach to be applied to aircraft collision avoidance. This path is optimized with respect to both fuel and/or time, and must ensure that the vehicles do not collide with each other or with obstacles, which can be fixed or moving according to a predefined motion.

An extension to this method is given by Richards and How in [32]. This research was driven by two major applications: air traffic management and Unmanned Aerial Vehicles (UAVs). According to this method, plume impingement constraints are also included in the MILP optimization problem.

1.3.5 Dynamic programming

Flint and Polycarpou in [18] present a stochastic decision process formulation for cooperative control among a team of distributed, uninhabited air vehicles (UAVs) which leads to a solution based on dynamic programming. The goal of each air vehicle in the search problem is to move over the environment such that, at the end of the search, the maximum amount of information about the environment has been gained by the team of vehicles. The path planning problem is discretized in time by allowing the vehicle to only make decisions at discrete time intervals.

According to Flint and Polycarpou dynamic programming provides a convenient modelling and analytical tool that many of the other, more intuitive or heuristic, approaches sometimes lack. For example, *dynamic programming can achieve a probably*

optimal global solution to the problem, at least in theory. In practice this is rarely the case, since *this may be computationally infeasible in the absence of further structural properties*, but dynamic programming can nevertheless serve as a blueprint for nearly optimal solutions.

A strict dynamic programming approach will have to compute to the very end of the vehicle's search the state, which would include every possible decision it could make. Since this state is a location and path on a map, it is natural to view this as a line extending behind the vehicle, connecting all the points the vehicle has travelled, and to view the planning of the vehicle as a tree starting from the vehicle's current location. This tree would then grow exponentially as the depth of the search (the number of steps ahead planned) increases. This quickly becomes intractable when the idealistic assumption of unlimited computation is removed.

Flint and Polycarpou use some simplified assumptions, such as a limited look-ahead policy. However, the introduction of the limited look-ahead policy can produce a suboptimal solution, because of the now limited scope of the vehicle's planning. Adding utility functions to the cost function allows for reducing these so-called horizon problems.

Expanding the formulation to include the presence of other vehicles, each vehicle can no longer correctly calculate where the other vehicles are going to be, because of the limited look-ahead and the limited communication assumptions. Calculating probabilities about the next positions of the other vehicles planning tree is *impractical* in the face of the computational complexity requirements. According to Flint and Polycarpou, a possible solution could be the assumption that the probability distribution has some structure in space. This structure can be simplified into several spatial regions, where each region represents a certain fixed probability of the vehicle causing an interference.

1.3.6 Remarks

Summarizing, deriving optimal paths for a multi-vehicle system in an unknown environment is still an open issue. All the previously discussed optimization methods does not include efficiently the unknown uncertainties of the environment. For example, calculating probabilities of the location of the threats in the dynamic programming approach is computationally impractical, while in case of the mixed-integer-programming or coordination-variables approach is not possible. Moreover, methods that are based on Voronoi polygonal paths can include only known probability distributions of the threats that are not updated online.

Therefore, finding an optimization method that will provide both online inclusion of the environment's uncertainties and computational efficiency is a great challenge for control engineers.

1.4 Objective of the thesis

This thesis is a small contribution to the problem of multi-vehicle path planning in an adversarial environment. To this end, we explore the utility of linear programming for trajectory planning in multi-vehicle control systems with adversaries. Several complicated multi-vehicle tasks are considered which are versions of the "Cornell RoboFlag competition".

Roughly speaking, these tasks involve two teams of robots, defenders and attackers, whose objective is the minimization or maximization of the number of robots that infiltrate a protected area, respectively. In such a game, coordination is of vital importance since either defenders or attackers must accomplish their goal with the minimum possible losses or the maximum possible gain, respectively.

However, at the same time prediction of the opponent's move and its inclusion to the optimization problem is necessary for computing efficient paths. Under these

conditions, this method could be really useful for efficient path planning in any multi-vehicle control task that include adversaries or unknown uncertainties.

1.5 Thesis outline

In Chapter 2, we construct a discrete-time state-space representation for the evolution of resources in an arena of sectors, which is a linear system. This model is commonly used in battlefield management, where proper arrangement of friendly resources is important.

In Chapter 3, the state-space representation of the resource allocation model is expanded in order to include the case that adversarial resources are also evolve in the same arena of sectors. We model enemy resources to follow similar state space equations as those of friendly resources, since they satisfy the same constraints as the friendly resources. We transform the problem of maximizing the enemy attrition losses into a large-scale optimization problem. In particular, we show that this problem can be approximated as a linear programming optimization problem and implemented in a receding horizon manner.

In Chapter 4, we explore the utility of the linear-programming-based planning for friendly resources allocation, described in Chapter 3, in deriving optimal paths for multi-vehicle control systems with adversaries. We consider the RoboFlag competition which involves two teams of robots with opposing interests, the defenders and the attackers. We derive control algorithms for both defense and attack that are based on the linear-programming-based planning for resource allocation. In this case, both defenders and attackers are modelled as different resource types in an arena of sectors. Moreover, since adversaries are modelled as state-dependent, various stochastic feedback laws based on their current position and/or velocity could describe their possible future attitude, which is included in the optimization.

In Chapter 5, the linear-programming-based multi-vehicle path planning for defense and attack derived for the RoboFlag competition in Chapter 4 is applied to the "Cornell RoboFlag Simulator" and to a RoboFlag simulator created in Matlab. We design several versions of the original RoboFlag competition and we test these control algorithms for both their effectiveness and computational efficiency.

CHAPTER 2

Resource Allocation Model

2.1 Introduction

In this chapter, we construct a model that describes the evolution of several resources in an arena of sectors. This model is motivated by the "SEADy storm" paradigm, which is described by Kott in [23]. It describes the movement and engagement of friendly and enemy forces which are represented as various resource types. According to this model, the battlefield is divided into a collection of sectors, while actions are undertaken in a discrete-time manner.

The state of the model consists of the level of a particular resource type per sector. It turns out that this model is a large-scale linear flow subject to various positivity constraints. Our final objective is to use this model for describing situations where friendly and enemy resources are engaged in an arena of sectors. The reason for this choice is grounded on the linearity of this model that allows for using linear objective functions in an optimization problem for friendly planning.

2.2 State-space representation

2.2.1 Selection of the states

Let us consider the simple case of an arena of only two sectors, s_1 and s_2, see Figure 2.1.

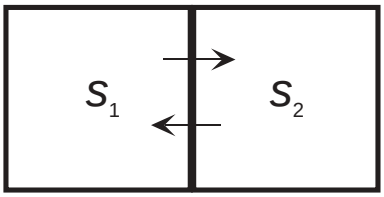


Figure 2.1: Arena of two sectors.

Assume that there are two types of resources, r_1 and r_2, which can move from one sector to another. We will also assume that each resource type cannot be transformed to another resource type, i.e., resource type r_1 cannot be transformed to resource type r_2 and vice versa.

The level of each resource type per sector describes the current state of the overall system. Accordingly, we define as *state vector* of this system to be the vector whose entries are the levels of resource types per sector. The entries of the *state vector* will be called *states*. In other words, the state vector of this system is

$$\mathbf{x} = \begin{pmatrix} x_{s_1,r_1} & x_{s_2,r_1} & x_{s_1,r_2} & x_{s_2,r_2} \end{pmatrix}^T \tag{2.1}$$

where x_{s_i,r_j} is the level of resource type r_j at sector s_i.

Similarly, for any number of sectors, n_s, and any of resources, n_r, the total number of states is $n_x = n_s n_r$, and the state vector is

$$\mathbf{x} = \begin{pmatrix} x_{s_1,r_1} & x_{s_2,r_1} & \dots & x_{s_{n_s},r_1} & \dots & x_{s_1,r_{n_r}} & x_{s_2,r_{n_r}} & \dots & x_{s_{n_s},r_{n_r}} \end{pmatrix}^T \in \Re_+^{n_x} \tag{2.2}$$

or

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_{r_1}^T & \mathbf{x}_{r_2}^T & \dots & \mathbf{x}_{r_{n_r}}^T \end{pmatrix}^T \in \Re_+^{n_x} \tag{2.3}$$

where

$$\mathbf{x}_{r_j} = \begin{pmatrix} x_{s_1,r_j} & x_{s_2,r_j} & \dots & x_{s_{n_s},r_j} \end{pmatrix}^T \in \Re_+^{n_s} \tag{2.4}$$

is the state vector of resource type r_j.

2.2.2 State evolution

The levels of each resource type are transferred in a discrete-time manner. They can either remain in the same sector or move to a neighboring sector, while movements within a sector are not taken into account. Therefore, the control action includes the transitions of resource levels from each sector to neighboring sectors.

Define $u_{s_i \leftarrow s_k, r_j}$ as the level of resource type r_j that is transferred from sector s_k to sector s_i, where s_k and s_i are neighboring sectors. Let also $x^+_{s_i,r_j}$ be the level of resource type r_j at sector s_i at the next stage. Then the system evolves according to the following state-space equation:

$$x^+_{s_i,r_j} = x_{s_i,r_j} + \sum_{k \in \mathcal{SN}(s_i,r_j)} u_{s_i \leftarrow s_k, r_j} - \sum_{k \in \mathcal{SN}(s_i,r_j)} u_{s_k \leftarrow s_i, r_j}, \quad s_i \in \mathcal{S}, \quad r_j \in \mathcal{R} \tag{2.5}$$

where $\mathcal{SN}(s_i, r_j)$ is the set of neighboring sectors of sector s_i that can be reached by resource type r_j in one stage, $\mathcal{S}$ is the set of sectors in the arena, and $\mathcal{R}$ is the set of resource types in the arena.

Define the control vector $\mathbf{u}$ as

$$\mathbf{u} = \begin{pmatrix} \mathbf{u}^T_{r_1} & \mathbf{u}^T_{r_2} & \dots & \mathbf{u}^T_{r_{n_r}} \end{pmatrix}^T \in \Re^{n_u}_+ \tag{2.6}$$

where the total number of control inputs is $n_u = n_r n_s (n_s - 1)$, and $\mathbf{u}_{r_j}$ is the control vector composed of the transitions of resource type r_j in the arena of sectors and defined by

$$\mathbf{u}_{r_j} = \begin{pmatrix} \mathbf{u}^T_{s_1,r_j} & \mathbf{u}^T_{s_2,r_j} & \dots & \mathbf{u}^T_{s_{n_s},r_j} \end{pmatrix}^T \in \Re^{n_s(n_s-1)}_+, \quad r_j \in \mathcal{R} \tag{2.7}$$

where $\mathbf{u}_{s_i,r_j}$ is the control vector that contains only the levels of resource type r_j that enters sector s_i, i.e.,

$$\begin{aligned} \mathbf{u}_{s_i,r_j} = \begin{pmatrix} u_{s_i \leftarrow s_1, r_j} & u_{s_i \leftarrow s_2, r_j} & \dots & u_{s_i \leftarrow s_{i-1}, r_j} & u_{s_i \leftarrow s_{i+1}, r_j} \dots & u_{s_i \leftarrow s_{n_s}, r_j} \end{pmatrix}^T \\ \in \Re^{(n_s-1)}_+, \quad s_i \in \mathcal{S}, \quad r_j \in \mathcal{R} \end{aligned} \tag{2.8}$$

Also define

$$\mathcal{U}_{\text{in}} : \mathcal{S} \times \mathcal{R} \times \mathcal{SN}(\mathcal{S}, \mathcal{R}) \to \{1, 2, \ldots, n_u\} \tag{2.9}$$

as the function that maps a triple of the form (s_i, r_j, s_k), where $s_i \in \mathcal{S}$, $r_j \in \mathcal{R}$ and $s_k \in \mathcal{SN}(s_i, r_j)$, to the row number of $\mathbf{u}$ that corresponds to $u_{s_i \leftarrow s_k, r_j}$, i.e.,

$$\mathbf{u}\left(\mathcal{U}_{\text{in}}\left(s_i, r_j, s_k\right), 1\right) = u_{s_i \leftarrow s_k, r_j} \tag{2.10}$$

Similarly, define

$$\mathcal{U}_{\text{out}} : \mathcal{SN}(\mathcal{S}, \mathcal{R}) \times \mathcal{R} \times \mathcal{S} \to \{1, 2, \ldots, n_u\} \tag{2.11}$$

as the function that maps a triple of the form (s_k, r_j, s_i), where $s_i \in \mathcal{S}$, $r_j \in \mathcal{R}$ and $s_k \in \mathcal{SN}(s_i, r_j)$, to the row number of $\mathbf{u}$ that corresponds to $u_{s_k \leftarrow s_i, r_j}$, i.e.,

$$\mathbf{u}\left(\mathcal{U}_{\text{out}}\left(s_k, r_j, s_i\right), 1\right) = u_{s_k \leftarrow s_i, r_j} \tag{2.12}$$

We can use $\mathcal{U}_{in}$ and $\mathcal{U}_{out}$ to rewrite the state-space equations of (2.5) in the following form:

$$x^{+}_{s_i, r_j} = x_{s_i, r_j} + \mathbf{b}^{T}_{s_i, r_j, \text{in}} \cdot \mathbf{u} - \mathbf{b}^{T}_{s_i, r_j, \text{out}} \cdot \mathbf{u}, \quad s_i \in \mathcal{S}, \quad r_j \in \mathcal{R} \tag{2.13}$$

where $\forall \theta \in \{1, 2, \ldots, n_u\}$ the following hold:

$$\mathbf{b}_{s_i, r_j, \text{in}}\left(\theta, 1\right) = \begin{cases} 1, & \text{if } \ \theta = \mathcal{U}_{\text{in}}\left(s_i, r_j, s_k\right), \quad s_k \in \mathcal{SN}\left(s_i, r_j\right) \\ 0, & \text{otherwise} \end{cases}$$

$$\mathbf{b}_{s_i, r_j, \text{out}}\left(\theta, 1\right) = \begin{cases} 1, & \text{if } \ \theta = \mathcal{U}_{\text{out}}\left(s_k, r_j, s_i\right), \quad s_k \in \mathcal{SN}\left(s_i, r_j\right) \\ 0, & \text{otherwise} \end{cases}$$

In other words, the inner product $\mathbf{b}^{T}_{s_i, r_j, \text{in}} \cdot \mathbf{u}$ is the level of resource type r_j that enters sector s_i, and the inner product $\mathbf{b}^{T}_{s_i, r_j, \text{out}} \cdot \mathbf{u}$ is the level of resource type r_j that exits from sector s_i.

Let us also define the following matrices:

$$\begin{aligned}\mathbf{B}_{\text{in}} &= \left[\begin{array}{cccccccc} \mathbf{b}_{s_1,r_1,\text{in}} & \dots & \mathbf{b}_{s_{n_s},r_1,\text{in}} & \dots & \mathbf{b}_{s_1,r_{n_r},\text{in}} & \dots & \mathbf{b}_{s_{n_s},r_{n_r},\text{in}} \end{array}\right]^T \\ \mathbf{B}_{\text{out}} &= \left[\begin{array}{cccccccc} \mathbf{b}_{s_1,r_1,\text{out}} & \dots & \mathbf{b}_{s_{n_s},r_1,\text{out}} & \dots & \mathbf{b}_{s_1,r_{n_r},\text{out}} & \dots & \mathbf{b}_{s_{n_s},r_{n_r},\text{out}} \end{array}\right]^T \end{aligned} \tag{2.14}$$

whose product with $\mathbf{u}$ is the vector of resource levels that enters or exits from each sector, respectively.

Then equation (2.13) can be written as

$$\mathbf{x}^+ = \mathbf{x} + (\mathbf{B}_{\text{in}} - \mathbf{B}_{\text{out}}) \cdot \mathbf{u} \tag{2.15}$$

or equivalently

$$\mathbf{x}^+ = \mathbf{x} + \mathbf{B} \cdot \mathbf{u} \tag{2.16}$$

where $\mathbf{x}^+$ is the state vector at the next stage and

$$\mathbf{B} = \mathbf{B}_{\text{in}} - \mathbf{B}_{\text{out}} \tag{2.17}$$

is the matrix whose product with $\mathbf{u}$ is the change in the state vector at the next stage. This matrix can also be given by the following equation:

$$\mathbf{B} = \left[\begin{array}{cccccccc} \mathbf{b}_{s_1,r_1} & \mathbf{b}_{s_2,r_1} & \dots & \mathbf{b}_{s_{n_s},r_1} & \dots & \mathbf{b}_{s_1,r_{n_r}} & \mathbf{b}_{s_2,r_{n_r}} & \dots & \mathbf{b}_{s_{n_s},r_{n_r}} \end{array}\right]^T \tag{2.18}$$

where

$$\mathbf{b}_{s_i,r_j} = \mathbf{b}_{s_i,r_j,\text{in}} - \mathbf{b}_{s_i,r_j,\text{out}}, \quad s_i \in \mathcal{S}, \quad r_j \in \mathcal{R} \tag{2.19}$$

is the vector whose inner product with $\mathbf{u}$ is the change in the level of resource type r_j at sector s_i.

For example, in case of two sectors and two resource types we have:

$$\mathcal{S} = \{s_1, s_2\}$$

$$\mathcal{R} = \{r_1, r_2\}$$

$$\mathcal{SN}(s_1, r_j) = \{s_2\}, \quad \forall r_j \in \mathcal{R}$$

$$\mathcal{SN}(s_2, r_j) = \{s_1\}, \quad \forall r_j \in \mathcal{R}$$

and the state-space equations are

$$\begin{pmatrix} x_{s_1,r_1} \\ x_{s_2,r_1} \\ x_{s_1,r_2} \\ x_{s_2,r_2} \end{pmatrix}^{+} = \begin{pmatrix} x_{s_1,r_1} \\ x_{s_2,r_1} \\ x_{s_1,r_2} \\ x_{s_2,r_2} \end{pmatrix} + \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} u_{s_1 \leftarrow s_2,r_1} \\ u_{s_2 \leftarrow s_1,r_1} \\ u_{s_1 \leftarrow s_2,r_2} \\ u_{s_2 \leftarrow s_1,r_2} \end{pmatrix} \tag{2.20}$$

2.2.3 State and control constraints

The state-space equation of (2.16) must meet several constraints in order to represent the evolution of the resources. In particular, the level x_{s_i,r_j} of resource type r_j at sector s_i can be neither less than zero nor greater than a positive upper bound, $x_{s_i,r_j,\max}$ (*positivity constraints*), i.e.,

$$0 \leq x_{s_i,r_j} \leq x_{s_i,r_j,\max},\ \forall s_i \in \mathcal{S},\ \forall r_j \in \mathcal{R} \tag{2.21}$$

These constraints can also be written in a vector form as

$$0 \leq \mathbf{x} \leq \mathbf{x}_{\max} \tag{2.22}$$

where $\mathbf{x}_{\max} \in \Re_{+}^{n_x}$ is the state vector of the maximum resource levels. Let us define $\mathcal{P}$ as the set of state vectors that satisfy the positivity constraints, i.e.,

$$\mathcal{P} = \{\mathbf{z} \in \Re_{+}^{n_x} \ \mid\ 0 \leq \mathbf{z} \leq \mathbf{x}_{\max}\} \tag{2.23}$$

Then $\mathbf{x}$ must satisfy the following constraint:

$$\mathbf{x} \in \mathcal{P} \tag{2.24}$$

Similarly, the control input $u_{s_k \leftarrow s_i,r_j}$ must always be non-negative, i.e.,

$$0 \leq u_{s_k \leftarrow s_i,r_j},\ \forall s_k \in \mathcal{SN}(s_i, r_j),\ \forall s_i \in \mathcal{S},\ \forall r_j \in \mathcal{R} \tag{2.25}$$

or equivalently

$$0 \leq \mathbf{u} \tag{2.26}$$

Moreover, the resources flow within the arena of sectors must be continuous, in the sense that the level of resource type r_j that exits from sector s_k cannot be greater than the level of resource type r_j that lies in sector s_k (*continuity constraints*), i.e.,

$$0 \leq \sum_{s_k \in \mathcal{SN}(s_i, r_j)} u_{s_k \leftarrow s_i, r_j} \leq x_{s_i, r_j}, \ \forall s_i \in \mathcal{S}, \ \forall r_j \in \mathcal{R} \tag{2.27}$$

which can be written as

$$0 \leq \mathbf{b}_{s_i, r_j, \text{out}}^T \cdot \mathbf{u} \leq x_{s_i, r_j}, \ \forall s_i \in \mathcal{S}, \ \forall r_j \in \mathcal{R} \tag{2.28}$$

or equivalently

$$\mathbf{0} \leq \mathbf{B}_{\text{out}} \cdot \mathbf{u} \leq \mathbf{x} \tag{2.29}$$

In addition, in order for constraint (2.22) to be satisfied, the control vector, $\mathbf{u}$, must also satisfy the following inequality:

$$\mathbf{0} \leq \mathbf{x}^+ \leq \mathbf{x}_{\max} \quad \Leftrightarrow \quad -\mathbf{x} \leq \mathbf{B} \cdot \mathbf{u} \leq \mathbf{x}_{\max} - \mathbf{x} \tag{2.30}$$

Hence, given current state vector $\mathbf{x}$, which satisfies the inequality constraint $\mathbf{0} \leq \mathbf{x} \leq \mathbf{x}_{\max}$, the control vector, $\mathbf{u}$, must be such that the following inequalities hold together:

$$\begin{cases} \mathbf{0} \leq \mathbf{u} \\ \mathbf{0} \leq \mathbf{B}_{\text{out}} \cdot \mathbf{u} \leq \mathbf{x} \\ -\mathbf{x} \leq \mathbf{B} \cdot \mathbf{u} \leq \mathbf{x}_{\max} - \mathbf{x} \end{cases} \tag{2.31}$$

Let us define $\mathcal{C}(\mathbf{x})$ as the set of control vectors that satisfy these constraints given current state vector $\mathbf{x}$, i.e.,

$$\mathcal{C}(\mathbf{x}) = \{\mathbf{z} \in \Re_+^{n_u} \ \mid \ \mathbf{B}_{\text{out}} \cdot \mathbf{z} \leq \mathbf{x}, \ \ -\mathbf{x} \leq \mathbf{B} \cdot \mathbf{z} \leq \mathbf{x}_{\max} - \mathbf{x}\} \tag{2.32}$$

Then $\mathbf{u}$ must satisfy the following constraint:

$$\mathbf{u} \in \mathcal{C}(\mathbf{x}) \tag{2.33}$$

Constraints (2.24) and (2.33) guarantee that the system is *closed*. In other words, the total quantity of a resource type in the arena of sectors cannot be increased or decreased, while resources can be moved only to neighboring sectors.

The state-space equation of (2.16) in conjunction with constraints (2.24) and (2.33) describe the evolution of resources flow. This system is given by

$$\Sigma_1 : \begin{cases} \mathbf{x}^+ = \mathbf{x} + \mathbf{B} \cdot \mathbf{u} \\ \\ \mathbf{x} \in \mathcal{P}, \quad \mathbf{u} \in \mathcal{C}(\mathbf{x}) \end{cases} \tag{2.34}$$

and can be represented by the block diagram in Figure 2.2.

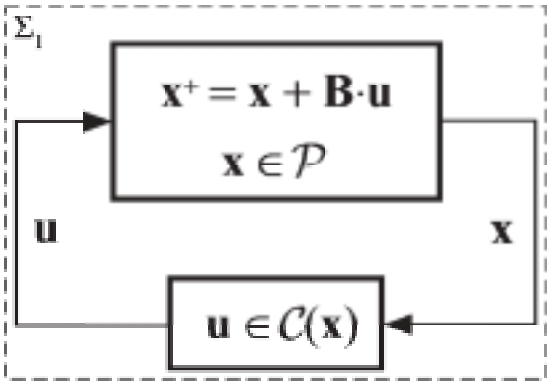


Figure 2.2: Block diagram of resources flow.

2.3 System specifications

According to the state-space representation of (2.34), both state vector $\mathbf{x}$ and control vector $\mathbf{u}$ are bounded by finite vectors. In particular,

$$\mathbf{0} \leq \mathbf{x} \leq \mathbf{x}_{\max} \tag{2.35}$$

and

$$-\mathbf{x}_{\max} \leq -\mathbf{x} \leq \mathbf{B} \cdot \mathbf{u} \leq \mathbf{x}_{\max} - \mathbf{x} \leq \mathbf{x}_{\max} \tag{2.36}$$

which means that Conclusion 2.1 holds.

Conclusion 2.1 *System (2.34) is bounded-input bounded-output stable.*

Moreover, in case we consider the sum of resource levels in the arena of sectors as the output of this system, i.e.,

$$\mathbf{y} = \mathbf{1}^T \cdot \mathbf{x} \tag{2.37}$$

the resulting system is

$$\begin{cases} \mathbf{x}^+ &= \mathbf{x} + \mathbf{B} \cdot \mathbf{u} \\ \mathbf{y} &= \mathbf{1}^T \cdot \mathbf{x} \end{cases} \tag{2.38}$$

and its transfer function is

$$\hat{\mathbf{G}}(z) = \mathbf{1}^T(z\mathbf{I} - \mathbf{I})^{-1}\mathbf{B} = \frac{1}{z-1}\mathbf{1}^T\mathbf{B} \tag{2.39}$$

where z is the z-transform variable. According to the definition of matrix $\mathbf{B}$, we have $\mathbf{1}^T\mathbf{B} = \mathbf{0}$. Hence,

$$\hat{\mathbf{G}}(z) = \mathbf{0} \tag{2.40}$$

which means that the output of this system is constant over time, i.e., Conclusion 2.2 holds.

Conclusion 2.2 *The sum of resource levels in the arena of sectors is constant over time.*

Since the sum of resource levels is constant over time, we can define the set $\mathcal{X}(\mathbf{x})$ of reachable state vectors given current state vector $\mathbf{x}$ as

$$\mathcal{X}(\mathbf{x}) = \{\mathbf{z} \in \mathcal{P} \ \mid \ \mathbf{1}^T \cdot \mathbf{z} = \mathbf{1}^T \cdot \mathbf{x}, \ \text{ where } \ \mathbf{x} \in \mathcal{P}\} \tag{2.41}$$

Let us also assume that every sector in the arena can be reached by every resource type, i.e.,

$$\mathcal{SN}(s_i, r_j) = \mathcal{SN}(s_i), \quad \forall r_j \in \mathcal{R} \tag{2.42}$$

where $\mathcal{SN}(s_i)$ is the set of neighboring sectors of sector s_i. Then, Conclusion 2.3 gives the conditions for a state vector to be reachable.

Conclusion 2.3 *Let* $\mathbf{x}[0] = \mathbf{x}_0$ *be an initial state vector of system (2.34) and* $\mathbf{x}_d$ *be any vector in* $\mathcal{P}$ *such that* $\mathbf{x}_d \in \mathcal{X}(\mathbf{x}_0)$. *If (2.42) is satisfied, then there is* $N \in \mathcal{N}$ *and an admissible control sequence* $\{\mathbf{u}[k]\}_{k=0}^{N}$, *with* $\mathbf{u}[k] \in \mathcal{C}(\mathbf{x}[k])$, $k \in \{0, 1, \ldots, N\}$, *such that* $\mathbf{x}[N] = \mathbf{x}_d$.

where $\mathcal{N}$ is the set of natural numbers.

2.4 Remarks

We conclude that resources flow in an arena of sectors can be described by the large scale linear system of equation (2.34). The sum of resource levels is always constant. If, in addition, any sector in the arena can be reached by any resource type, i.e., equation (2.42) is satisfied, then the system can be driven to any desired state within the set of allowable states.

In the next chapter, this linear system with constraints will also be the basis of a model for an adversarial team that allocates its resources in the same arena of sectors. This model will be used in describing situations where friendly and enemy resources are engaged in an arena of sectors. Moreover, because of the linearity of this model we will show that a linear-programming-based planning can be used for friendly resources allocation.

CHAPTER 3

Adversarial Environment

3.1 Introduction

In this chapter, the linear system of equation (2.34), which describes the evolution of resources in an arena of sectors, is expanded in order to include the case that adversarial resources are also evolve in the same arena of sectors. This situation is appropriate in battles, where several resource types of two teams are engaged trying to cause the largest possible attrition to their enemies. In this case, attrition occurs when two resource types of different team are very close to each other, or if they lie in the same sector.

Each team allocates its resources in such a way that opponent's attrition losses are maximized. Several approaches to deriving optimal decisions for the "friendly" resources (home team) have been already proposed. One of these is to view the presence of the enemy resources (opponent team) as a "disturbance" and to design optimal disturbance rejection controllers. In this case, both control and disturbance must satisfy constraints similar to those of equation (2.33) since they cannot be greater than the available resources and lower than zero. Methods such as ℓ_1 optimal control [10] do take into account the possibility of such bounds.

However, since enemy resources are also state dependent, we can avoid taking a conventional disturbance rejection approach. In particular, we will model enemy resources to follow similar state space equations as those of friendly resources, since they evolve following the same constraints (positivity and continuity constraints).

We also assume that the decisions of both teams can be made within the same time-interval and that the current state of the enemy resources is known.

According to these assumptions, we will transform the problem of maximizing the enemy attrition losses into a large-scale optimization problem. In particular, we will show that this problem can be approximated as a linear program, and implemented in a receding horizon manner. This approach was introduced by Daniel-Berhe et al. in [13] and will be the basis for a multi-vehicle path planning with adversaries in Chapter 4.

3.2 State-space representation

Let us consider the case where resources are subject to attrition because in the arena of sectors another team of resources (adversarial resources) is evolved. Define "*friendly team*" to be the home team of resources, which will be denoted by the letter "f", and define the team of adversarial resources as the "*enemy team*", denoted by the letter "e". We will assume that the set of resource types, $\mathcal{R}$, is the same for each team, i.e.,

$$\mathcal{R}^f = \mathcal{R}^e = \mathcal{R} \tag{3.1}$$

which also implies that

$$\mathcal{SN}^f(s_i, r_j) = \mathcal{SN}^e(s_i, r_j) = \mathcal{SN}(s_i, r_j), \quad \forall s_i \in \mathcal{S}, \quad \forall r_j \in \mathcal{R} \tag{3.2}$$

In this case, *attrition* is defined as the level of a resource type of either team that is being lost when it meets the same resource type of the opponent team. In particular, suppose that at sector s_i the level of friendly resource type r_j is $x^f_{s_i,r_j}$, while the level of the enemy resource type r_j is $x^e_{s_i,r_j}$. Then, for both teams the attrition function for resource type r_j at sector s_i will be

$$l_{s_i,r_j} = l_{s_i,rj}\left(x^f_{s_i,r_j}, x^e_{s_i,r_j}\right) = \min\left\{x^f_{s_i,r_j}, x^e_{s_i,r_j}\right\} \tag{3.3}$$

In case there is no alignment of resources, then the attrition loss will be equal to zero.

Because of the attrition losses, the states of friendly and enemy resources are

$$\left(x^f_{s_i,r_j}\right)^+ = x^f_{s_i,r_j} + \sum_{k\in\mathcal{SN}(s_i,r_j)} u^f_{s_i\leftarrow s_k,r_j} - \sum_{k\in\mathcal{SN}(s_i,r_j)} u^f_{s_k\leftarrow s_i,r_j} - l^f_{s_i,r_j} \tag{3.4}$$

$$\left(x^e_{s_i,r_j}\right)^+ = x^e_{s_i,r_j} + \sum_{k\in\mathcal{SN}(s_i,r_j)} u^e_{s_i\leftarrow s_k,r_j} - \sum_{k\in\mathcal{SN}(s_i,r_j)} u^e_{s_k\leftarrow s_i,r_j} - l^e_{s_i,r_j} \tag{3.5}$$

$$\forall s_i \in \mathcal{S}, \quad \forall r_j \in \mathcal{R}$$

where

$$l^f_{s_i,r_j} = l^e_{s_i,r_j} = l_{s_i,rj}\left(x^f_{s_i,r_j}, x^e_{s_i,r_j}\right)$$

Equations (3.4), (3.5) can be written equivalently as

$$\begin{pmatrix} \mathbf{x}^f \\ \mathbf{x}^e \end{pmatrix}^+ = \begin{pmatrix} \mathbf{x}^f \\ \mathbf{x}^e \end{pmatrix} + \begin{pmatrix} \mathbf{B}^f \\ \mathbf{O} \end{pmatrix} \cdot \mathbf{u}^f + \begin{pmatrix} \mathbf{O} \\ \mathbf{B}^e \end{pmatrix} \cdot \mathbf{u}^e - \begin{pmatrix} \mathbf{l}^f \\ \mathbf{l}^e \end{pmatrix} \tag{3.6}$$

$$\mathbf{x}^f, \mathbf{x}^e \in \Re^{n_x}_+, \quad \mathbf{u}^f, \mathbf{u}^e \in \Re^{n_u}_+, \quad \mathbf{B}^f, \mathbf{B}^e \in \Re^{n_x \times n_u}$$

where

$$\mathbf{l}^f = \begin{pmatrix} l^f_{s_1,r_1} & l^f_{s_2,r_1} & \dots & l^f_{s_{n_s},r_1} & \dots & l^f_{s_1,r_{n_r}} & l^f_{s_2,r_{n_r}} & \dots & l^f_{s_{n_s},r_{n_r}} \end{pmatrix}^T \in \Re^{n_x}$$

$$\mathbf{l}^e = \begin{pmatrix} l^e_{s_1,r_1} & l^e_{s_2,r_1} & \dots & l^e_{s_{n_s},r_1} & \dots & l^e_{s_1,r_{n_r}} & l^e_{s_2,r_{n_r}} & \dots & l^e_{s_{n_s},r_{n_r}} \end{pmatrix}^T \in \Re^{n_x}$$

Because of (3.1) in the above equations we have also assumed that

$$n^f_x = n^e_x = n_x \quad \text{and} \quad n^f_u = n^e_u = n_u \tag{3.7}$$

In order for the state-space equations of (3.6) to represent the flow of both friendly and enemy resources, constraints (2.24) and (2.33) must also be taken into account. In particular, the resources flow is described by

$$\Sigma_2 : \quad \begin{cases} \begin{pmatrix} \mathbf{x}^f \\ \mathbf{x}^e \end{pmatrix}^+ = \begin{pmatrix} \mathbf{x}^f \\ \mathbf{x}^e \end{pmatrix} + \begin{pmatrix} \mathbf{B}^f \\ \mathbf{O} \end{pmatrix} \cdot \mathbf{u}^f + \begin{pmatrix} \mathbf{O} \\ \mathbf{B}^e \end{pmatrix} \cdot \mathbf{u}^e - \begin{pmatrix} \mathbf{l}^f \\ \mathbf{l}^e \end{pmatrix} \\ \mathbf{x}^f, \mathbf{x}^e \in \mathcal{P}, \quad \mathbf{u}^f \in \mathcal{C}(\mathbf{x}^f), \quad \mathbf{u}^e \in \mathcal{C}(\mathbf{x}^e) \end{cases} \tag{3.8}$$

where the sets $\mathcal{P}$ and $\mathcal{C}(\mathbf{x})$ are defined by (2.23), (2.32), respectively.

The evolution of friendly and enemy resources can also be represented by the block diagram in Figure 3.1.

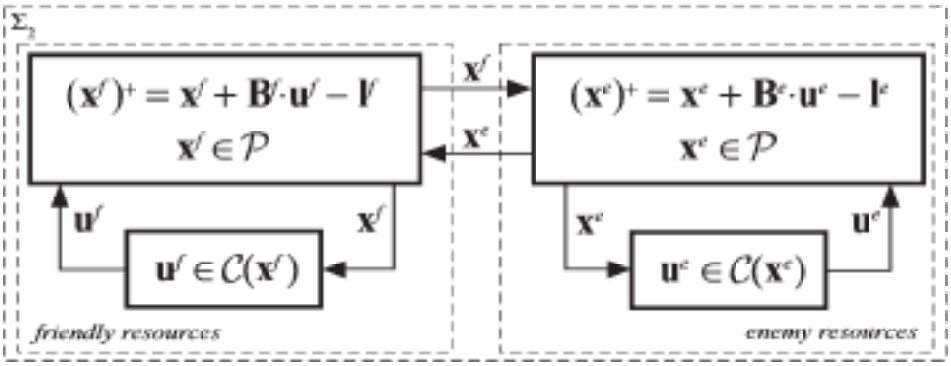


Figure 3.1: Block diagram of friendly and enemy resources flow with attrition.

3.3 Problem reformulation

Our goal is to maximize the attrition of the enemy, or equivalently to minimize its resource levels. In other words, we are searching for friendly paths that will cause the maximum possible attrition to the enemy.

Using the attrition function of (3.3), the state-space equation of (3.8) is not linear any more, because the attrition function is a nonlinear function of the state vectors of both teams. Nevertheless, even if the attrition function is a linear function of the enemy state vector, i.e.,

$$\mathbf{l}^f = \mathbf{L}^f \cdot \mathbf{x}^e, \quad \mathbf{l}^e = \mathbf{L}^e \cdot \mathbf{x}^f \tag{3.9}$$

the state-space equations of (3.8) may not satisfy the positivity constraint.

In particular, since the attrition losses are subtracted from the current state, if the current resource level at a sector is zero, then the corresponding resource level at the next stage will be negative. Thus, by using the state-space equations of (3.8) with

the additional constraints of (3.9), we are not able to formulate a linear optimization program, where the objective function would be the minimization of the enemy's attrition losses. A linear formulation is desirable because it is easy to solve and computationally efficient.

According to Daniel-Berhe et al in [13], this optimization can be approximated with a linear programming based planning for the friendly resources. This approach is based on two model simplifications that will allow the use of linear programming. To compensate for this approximation the optimization scheme will be implemented according to a receding horizon manner.

3.3.1 Model simplifications

The state-space equations of both teams (*friendly* and *enemy*), given by (3.8) in Section 3.1, cannot be used for deriving an optimal policy for friendly planning. The obstacles are

- The presence of the attrition function.
- The unknown control vector of the enemy resources.

For this reason, we remove the attrition function from the state-space equations, i.e.,

$$\begin{cases} \begin{pmatrix} \mathbf{x}^f \\ \mathbf{x}^e \end{pmatrix}^+ = \begin{pmatrix} \mathbf{x}^f \\ \mathbf{x}^e \end{pmatrix} + \begin{pmatrix} \mathbf{B}^f \\ \mathbf{O} \end{pmatrix} \cdot \mathbf{u}^f + \begin{pmatrix} \mathbf{O} \\ \mathbf{B}^e \end{pmatrix} \cdot \mathbf{u}^e \\ \mathbf{x}^f, \mathbf{x}^e \in \mathcal{P}, \quad \mathbf{u}^f \in \mathcal{C}(\mathbf{x}^f), \quad \mathbf{u}^e \in \mathcal{C}(\mathbf{x}^e) \end{cases} \tag{3.10}$$

In other words, we assume that both teams evolve as if no attrition will occur. However, we can expect that this model will be significantly different from the actual evolution. We overcome this problem by applying a receding horizon philosophy that will be described later in this chapter.

Moreover, since the control vector of enemy resources is not known to the friendly resources, we assume that enemy resources implements a known feedback policy $\mathbf{G}^e \in \Re^{n_u \times n_x}$, i.e.,

$$(\mathbf{x}^e)^+ = \mathbf{x}^e + \mathbf{B}^e \cdot (\mathbf{G}^e \mathbf{x}^e) = (\mathbf{I} + \mathbf{B}^e \mathbf{G}^e) \cdot \mathbf{x}^e \tag{3.11}$$

where $\mathbf{I} \in \Re^{n_x \times n_x}$ is the identity matrix.

In this case, the state-space representation for both friendly and enemy resources takes on the form

$$\Sigma_2' : \quad \begin{cases} \begin{pmatrix} \mathbf{x}^f \\ \mathbf{x}^e \end{pmatrix}^+ = \begin{pmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{I} + \mathbf{B}^e \mathbf{G}^e \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x}^f \\ \mathbf{x}^e \end{pmatrix} + \begin{pmatrix} \mathbf{B}^f \\ \mathbf{O} \end{pmatrix} \cdot \mathbf{u}^f \\ \mathbf{x}^f, \mathbf{x}^e \in \mathcal{P}, \quad \mathbf{u}^f \in \mathcal{C}(\mathbf{x}^f), \quad \mathbf{G}^e \mathbf{x}^e \in \mathcal{C}(\mathbf{x}^e) \end{cases} \tag{3.12}$$

This system can be described by the block diagram in Figure 3.2.

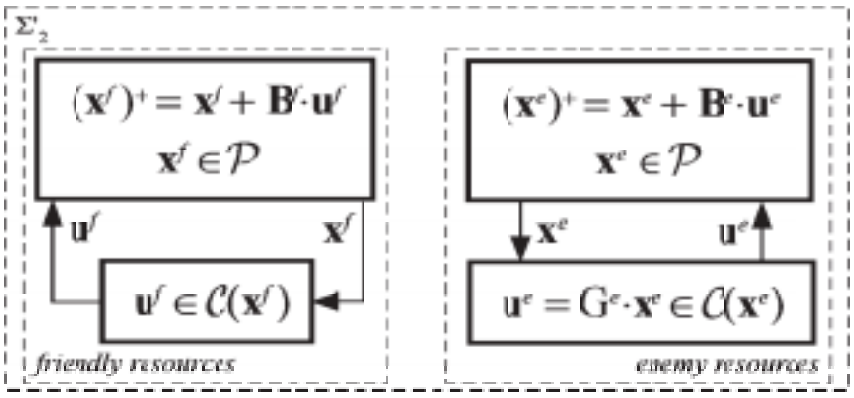


Figure 3.2: Block diagram of the simplified model for friendly and enemy resources flow.

By using this simplified model, friendly resources are able to decide about their future states without considering the case of possible attrition losses. However, that decision is based on a prediction of the enemy's future states, that are included in the feedback matrix $\mathbf{G}^e$.

3.3.2 Creation of the enemy's feedback matrix

The feedback matrix, $\mathbf{G}^e$, contains the assumed information about the future states of enemy resources. It is generally unknown to friendly resources but introduced for the sake of optimization. More specifically, this feedback matrix can be used for modelling

- certain paths of enemy resources
- diffusion of enemy resources
- probability maps of enemy resources

Recall that $\mathcal{U}_{\text{out}} : \mathcal{SN}(\mathcal{S}, \mathcal{R}) \times \mathcal{R} \times \mathcal{S} \to \{1, 2, \ldots, n_u\}$ is the function that maps a triple of the form (s_k, r_j, s_i), where $s_i \in \mathcal{S}$, $r_j \in \mathcal{R}$ and $s_k \in \mathcal{SN}(s_i, r_j)$, to the row number of $\mathbf{u}$ that corresponds to $u_{s_k \leftarrow s_i, r_j}$. Similarly, in case of enemy resources we have

$$\mathbf{u}^e \left(\mathcal{U}_{\text{out}} \left(s_k, r_j, s_i \right), 1 \right) = u^e_{s_k \leftarrow s_i, r_j} \tag{3.13}$$

Then, for every resource type $r_j \in \mathcal{R}$, we define a feedback matrix $\mathbf{G}^e_{r_j} \in \Re^{n_u \times n_s}$, such that

$$\mathbf{G}^e_{r_j} \left(\mathcal{U}_{\text{out}} \left(s_k, r_j, s_i \right), s_i \right) = g^e_{s_k \leftarrow s_i, r_j} \tag{3.14}$$

where $g^e_{s_k \leftarrow s_i, r_j}$ is of our choice and satisfies the following properties:

$$\begin{aligned} &(1) \quad g^e_{s_k \leftarrow s_i, r_j} \in [0, 1] \\ &(2) \quad u^e_{s_k \leftarrow s_i, r_j} = g^e_{s_k \leftarrow s_i, r_j} \cdot x^e_{s_i, r_j} = g^e_{s_k \leftarrow s_i, r_j} \cdot \mathbf{x}^e_{r_j} \left(s_i, 1 \right) \\ &(3) \quad \textstyle\sum_{s_k \in \mathcal{SN}(s_i, r_j)} \left\{ g^e_{s_k \leftarrow s_i, r_j} \right\} \leq 1 \end{aligned} \tag{3.15}$$

These properties guarantee that the *control constraints* of (3.12) are satisfied. In other words, $g^e_{s_k \leftarrow s_i, r_j}$ is the percentage of level x_{s_i, r_j} of resource type $r_j \in \mathcal{R}$ at sector $s_i \in \mathcal{S}$, that will be transferred to sector $s_k \in \mathcal{SN}(s_i, r_j)$.

Recall,

$$\mathbf{x}^e = \left(\begin{array}{cccc} \left(\mathbf{x}^e_{r_1}\right)^T & \left(\mathbf{x}^e_{r_2}\right)^T & \ldots & \left(\mathbf{x}^e_{r_{n_r}}\right)^T \end{array} \right)^T \in \Re_+^{n_x} \tag{3.16}$$

and

$$\mathbf{u}^e = \left(\begin{array}{cccc} \left(\mathbf{u}^e_{r_1}\right)^T & \left(\mathbf{u}^e_{r_2}\right)^T & \ldots & \left(\mathbf{u}^e_{r_{n_r}}\right)^T \end{array} \right)^T \in \Re_+^{n_u} \tag{3.17}$$

We can define the enemy's feedback matrix, $\mathbf{G}^e$, as

$$\mathbf{G}^e = \left[\begin{array}{cccc} \mathbf{G}^e_{r_1} & \mathbf{O} & \ldots & \mathbf{O} \\ \mathbf{O} & \mathbf{G}^e_{r_2} & \ldots & \mathbf{O} \\ \ldots & \ldots & \ldots & \ldots \\ \mathbf{O} & \mathbf{O} & \ldots & \mathbf{G}^e_{r_{n_r}} \end{array} \right] \tag{3.18}$$

where

$$\mathbf{u}^e_{r_j} = \mathbf{G}^e_{r_j} \cdot \mathbf{x}^e_{r_j}, \quad \forall r_j \in \mathcal{R} \tag{3.19}$$

or, equivalently,

$$\mathbf{u}^e = \mathbf{G}^e \cdot \mathbf{x}^e \tag{3.20}$$

According to the previous definition of feedback matrix $\mathbf{G}^e$, if we set $g^e_{s_k \leftarrow s_i, r_j} = 1$ or 0, then we define a *certain* next destination for resource type r_j, namely it will move from sector s_i to sector s_k or it will remain at sector s_i. Moreover, if we split the resource level to two or more destination sectors, then we create a *diffusion* of resources. This means $g^e_{s_k \leftarrow s_i, r_j} < 1$. Finally, $g^e_{s_k \leftarrow s_i, r_j}$ could be interpreted as the *probability* that resource type r_j will be moved from sector s_i to sector s_k.

The probabilistic interpretation of $\mathbf{G}^e$ is very useful, since we desire to test the effectiveness of friendly planning versus an unknown sequence of enemy decisions. Moreover, since $\mathbf{G}^e$ models a prediction about the future states of enemy resources, we are able to optimize the states of friendly resources for several stages ahead. In this way, we can compute optimal paths instead of optimal next states.

3.4 Optimization set-up

3.4.1 Objective function

The question that arises now is which objective function would be proper for friendly optimization planning. Generally speaking, the objective of the friendly resources is the maximization of the enemy's attrition losses. Since attrition occurs when both friendly and enemy resources lie in the same sector, a possible friendly objective could be

$$\text{minimize} \quad \left\|(\mathbf{x}^f)^+ - (\mathbf{x}^e)^+\right\|_{\ell_1} = \sum_{i=1}^{n_s} \left\{ \sum_{j=1}^{n_r} \left| (x^f_{s_i,r_j})^+ - (x^e_{s_i,r_j})^+ \right| \right\} \tag{3.21}$$

In other words, compute the next state of friendly resources in order to achieve the minimum possible difference with the next state of enemy resources. In this case, the attrition of enemy resources will be maximized.

By using such an objective function, the state of friendly resources is optimized for only the next stage. However, it could be the case that the predicted next positions of enemy resources are not within the one-stage reachable set of sectors of friendly resources. In this case, the optimal next friendly state would be equal to the current state. In other words, friendly resources will not move towards the enemy resources.

Certainly, the previous objective function does not guarantee an optimal path that will maximize the enemy's attrition losses. For this reason, and since we are able to create a stochastic feedback matrix $\mathbf{G}^e$ to model the enemy's routes for more than one stage ahead, we expand the previous objective function to N_p stages ahead. The parameter N_p is the optimization horizon and coincides with the prediction horizon for the future states of enemy resources.

Define the state vectors of both teams for the whole optimization horizon, given

by

$$\mathbf{X}^f_{1\to N_p} = \begin{pmatrix} \mathbf{x}^f[1] \\ \mathbf{x}^f[2] \\ \dots \\ \mathbf{x}^f[N_p] \end{pmatrix} \in \Re_+^{N_p n_x}, \quad \mathbf{X}^e_{1\to N_p} = \begin{pmatrix} \mathbf{x}^e[1] \\ \mathbf{x}^e[2] \\ \dots \\ \mathbf{x}^e[N_p] \end{pmatrix} \in \Re_+^{N_p n_x} \tag{3.22}$$

where $\mathbf{x}^f[t]$ and $\mathbf{x}^e[t]$ are the state vectors of friendly and enemy resources, respectively, at the t-th future stage.

The new objective function is

$$\text{minimize} \quad \|\mathbf{X}^f_{1\to N_p} - \mathbf{X}^e_{1\to N_p}\|_{\ell_1} \tag{3.23}$$

which equivalently can be written as

$$\text{minimize} \quad \sum_{t=1}^{N_p} \left\|\mathbf{x}^f[t] - \mathbf{x}^e[t]\right\|_{\ell_1} = \sum_{t=1}^{N_p} \left\{ \sum_{i=1}^{n_s} \left\{ \sum_{j=1}^{n_r} \left| x^f_{s_i,r_j}[t] - x^e_{s_i,r_j}[t] \right| \right\} \right\} \tag{3.24}$$

3.4.2 Dynamic constraints

The state vectors of both teams evolve according to the simplified model given by (3.12). For example, for any initial friendly state $\mathbf{x}^f[0]$, we have

$$\begin{aligned} \mathbf{x}^f[1] &= \mathbf{x}^f[0] + \mathbf{B}^f \cdot \mathbf{u}^f[0] \\ \mathbf{x}^f[2] &= \mathbf{x}^f[1] + \mathbf{B}^f \cdot \mathbf{u}^f[1] = \mathbf{x}^f[0] + \begin{bmatrix} \mathbf{B}^f & \mathbf{B}^f \end{bmatrix} \cdot \begin{pmatrix} \mathbf{u}^f[0] \\ \mathbf{u}^f[1] \end{pmatrix} \end{aligned}$$

and generally the state after N_p stages is

$$\begin{aligned} \mathbf{x}^f[N_p] &= \mathbf{x}^f[N_p - 1] + \mathbf{B}^f \cdot \mathbf{u}^f[N_p - 1] = \\ &= \mathbf{x}^f[0] + \begin{bmatrix} \mathbf{B}^f & \mathbf{B}^f & \dots & \mathbf{B}^f \end{bmatrix} \cdot \begin{pmatrix} \mathbf{u}^f[0] \\ \mathbf{u}^f[1] \\ \dots \\ \mathbf{u}^f[N_p - 1] \end{pmatrix} \end{aligned}$$

The state-space equations of friendly resources and for optimization horizon N_p can be written equivalently as

$$\mathbf{X}^f_{1\to N_p} = \mathbf{T}^f_{xx_0} \cdot \mathbf{x}^f[0] + \mathbf{T}^f_{xu} \cdot \mathbf{U}^f_{0\to(N_p-1)} \tag{3.25}$$

where

$$\mathbf{T}^f_{xx_0} = \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \\ \dots \\ \mathbf{I} \end{bmatrix} \in \Re^{N_p n_x \times n_x}, \quad \mathbf{T}^f_{xu} = \begin{bmatrix} \mathbf{B}^f & \mathbf{O} & \dots & \mathbf{O} \\ \mathbf{B}^f & \mathbf{B}^f & \dots & \mathbf{O} \\ \dots & \dots & \dots & \dots \\ \mathbf{B}^f & \mathbf{B}^f & \dots & \mathbf{B}^f \end{bmatrix} \in \Re^{N_p n_x \times N_p n_u}$$

and

$$\mathbf{U}^f_{0\to(N_p-1)} = \begin{pmatrix} \mathbf{u}^f[0] \\ \mathbf{u}^f[1] \\ \dots \\ \mathbf{u}^f[N_p-1] \end{pmatrix} \in \Re_+^{N_p n_u}$$

is the control vector of friendly resources for optimization horizon N_p.

Similarly, if the current state of enemy resources is $\mathbf{x}^e[0]$, then the states for the next two stages are

$$\begin{aligned} \mathbf{x}^e[1] &= (\mathbf{I} + \mathbf{B}^e\mathbf{G}^e) \cdot \mathbf{x}^e[0] \\ \mathbf{x}^e[2] &= (\mathbf{I} + \mathbf{B}^e\mathbf{G}^e) \cdot \mathbf{x}^e[1] = (\mathbf{I} + \mathbf{B}^e\mathbf{G}^e)^2 \cdot \mathbf{x}^e[0] \end{aligned}$$

and generally the state after N_p stages is

$$\mathbf{x}^e[N_p] = (\mathbf{I} + \mathbf{B}^e\mathbf{G}^e)^{N_p} \cdot \mathbf{x}^e[0]$$

The state-space equations of the enemy resources for the whole optimization horizon N_p can be written equivalently as

$$\mathbf{X}^e_{1\to N_p} = \mathbf{T}^e_{xx_0,G} \cdot \mathbf{x}^e[0] \tag{3.26}$$

where

$$\mathbf{T}^e_{xx_0,G} = \begin{bmatrix} (\mathbf{I} + \mathbf{B}^e\mathbf{G}^e) \\ (\mathbf{I} + \mathbf{B}^e\mathbf{G}^e)^2 \\ \dots \\ (\mathbf{I} + \mathbf{B}^e\mathbf{G}^e)^{N_p} \end{bmatrix} \in \Re^{N_p n_x \times n_x}$$

3.4.3 State constraints

According to the simplified model of (3.12), the state vectors of both teams must belong to $\mathcal{P}$. In other words, the resource level at any sector must always be non-negative, while it is bounded from above by a positive resource level. In particular,

as far as friendly resources are concerned, the following inequality must be satisfied:

$$0 \leq \mathbf{X}^f_{1\to N_p} \leq \mathbf{X}^f_{\max} \tag{3.27}$$

where

$$\mathbf{X}^f_{\max} = \mathbf{T}^f_{xx,\mathrm{id}} \cdot \mathbf{x}^f_{\max}$$

$$\mathbf{T}^f_{xx,\mathrm{id}} = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} \end{bmatrix}^T \in \Re^{N_p n_x \times n_x}$$

Similarly, for the enemy resources

$$0 \leq \mathbf{X}^e_{1\to N_p} \leq \mathbf{X}^e_{\max} \tag{3.28}$$

where

$$\mathbf{X}^e_{\max} = \mathbf{T}^e_{xx,\mathrm{id}} \cdot \mathbf{x}^e_{\max}$$

$$\mathbf{T}^e_{xx,\mathrm{id}} = \begin{bmatrix} \mathbf{I} & \mathbf{I} & \dots & \mathbf{I} \end{bmatrix}^T \in \Re^{N_p n_x \times n_x}$$

3.4.4 Control constraints

According to model (3.12), the allowable control input of friendly resources must belong to the set $\mathcal{C}(\mathbf{x}^f)$. In particular, the resource level that exits from a sector must be non-negative, i.e.,

$$\mathbf{U}^f_{0\to(N_p-1)} \geq 0 \tag{3.29}$$

while the resources flow must be continuous, in the sense that the resource level that exits from a sector must be less than the resource level at that sector. These constraints can be represented by the following inequalities:

$$\begin{cases} \mathbf{B}^f_{\mathrm{out}} \cdot \mathbf{u}^f[0] & \leq & \mathbf{x}^f[0] \\ \mathbf{B}^f_{\mathrm{out}} \cdot \mathbf{u}^f[1] & \leq & \mathbf{x}^f[0] \\ \dots & \dots & \dots \\ \mathbf{B}^f_{\mathrm{out}} \cdot \mathbf{u}^f[N_p-1] & \leq & \mathbf{x}^f[N_p-1] \end{cases} \tag{3.30}$$

which equivalently can be written as

$$\mathbf{T}^f_c \cdot \mathbf{U}^f_{0\to(N_p-1)} \leq \mathbf{X}^f_{0\to(N_p-1)} \tag{3.31}$$

where

$$\mathbf{T}_c^f = \begin{bmatrix} \mathbf{B}_{\text{out}}^f & \mathbf{O} & \dots & \mathbf{O} \\ \mathbf{O} & \mathbf{B}_{\text{out}}^f & \dots & \mathbf{O} \\ \dots & \dots & \dots & \dots \\ \mathbf{O} & \mathbf{O} & \dots & \mathbf{B}_{\text{out}}^f \end{bmatrix} \in \Re^{N_p n_x \times N_p n_u}, \quad \mathbf{X}_{0\to(N_p-1)} = \begin{pmatrix} \mathbf{x}^f[0] \\ \mathbf{x}^f[1] \\ \dots \\ \mathbf{x}^f[N_p-1] \end{pmatrix} \in \Re_+^{N_p n_x}$$

Similarly to (3.25), we have

$$\mathbf{X}_{0\to(N_p-1)}^f = \mathbf{T}_{xx_0,c} \cdot \mathbf{x}^f[0] + \mathbf{T}_{xu,c}^f \cdot \mathbf{U}_{0\to(N_p-1)}^f \tag{3.32}$$

where

$$\mathbf{T}_{xx_0,c}^f = \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \\ \dots \\ \mathbf{I} \end{bmatrix} \in \Re^{N_p n_x \times n_x}, \quad T_{xu,c}^f = \begin{bmatrix} \mathbf{O} & \mathbf{O} & \dots & \mathbf{O} & \mathbf{O} \\ \mathbf{B}^f & \mathbf{O} & \dots & \mathbf{O} & \mathbf{O} \\ \mathbf{B}^f & \mathbf{B}^f & \dots & \mathbf{O} & \mathbf{O} \\ \dots & \dots & \dots & \dots & \dots \\ \mathbf{B}^f & \mathbf{B}^f & \dots & \mathbf{O} & \mathbf{O} \\ \mathbf{B}^f & \mathbf{B}^f & \dots & \mathbf{B}^f & \mathbf{O} \end{bmatrix} \in \Re^{N_p n_x \times N_p n_u}$$

Thus, the *continuity constraints* of (3.31) can be written as

$$\mathbf{T}_c \cdot \mathbf{U}_{0\to(N_p-1)}^f \le \mathbf{T}_{xx_0,c} \cdot \mathbf{x}^f[0] + \mathbf{T}_{xu,c}^f \cdot \mathbf{U}_{0\to(N_p-1)}^f \tag{3.33}$$

or equivalently

$$\left(\mathbf{T}_c - \mathbf{T}_{xu,c}^f\right) \cdot \mathbf{U}_{0\to(N_p-1)}^f \le \mathbf{T}_{xx_0,c} \cdot \mathbf{x}^f[0] \tag{3.34}$$

Moreover, according to the definition of $\mathcal{C}(\mathbf{x}^f)$, the following control constraint must be satisfied:

$$-\mathbf{x}^f \le \mathbf{B}^f \cdot \mathbf{u}^f \le \mathbf{x}_{\max} - \mathbf{x}^f \tag{3.35}$$

or equivalently

$$\mathbf{0} \le (\mathbf{x}^+)^f \le \mathbf{x}_{\max} \tag{3.36}$$

However, these constraints can be ignored, since the state constraints of Section 3.4.3 include the state vectors for the whole optimization horizon, (3.27). Hence, the control constraints are given by (3.29) and (3.34).

3.4.5 Convex optimization problem

The objective function of (3.23) accompanied with the dynamic constraints of (3.25), the state constraints of (3.27) and the control constraints of (3.29), (3.34), form the following optimization problem for friendly planning:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{X}^f - \mathbf{X}^e\|_{\ell_1} \\ \text{subject to} \quad & \mathbf{X}^f = \mathbf{T}^f_{xx_0} \cdot \mathbf{x}^f[0] + \mathbf{T}^f_{xu} \cdot \mathbf{U}^f \\ & \mathbf{0} \leq \mathbf{X}^f \leq \mathbf{X}^f_{\max} \\ & \mathbf{0} \leq \mathbf{U}^f \\ & \left(\mathbf{T}_c - \mathbf{T}^f_{xu,c}\right) \cdot \mathbf{U}^f \leq \mathbf{T}_{xx_0,c} \cdot \mathbf{x}^f[0] \end{aligned} \tag{3.37}$$

where, for the sake of simplicity we have set

$$\mathbf{X}^f_{1\to N_p} = \mathbf{X}^f, \quad \mathbf{X}^e_{1\to N_p} = \mathbf{X}^e, \quad \mathbf{U}^f_{0\to(N_p-1)} = \mathbf{U}^f \tag{3.38}$$

The variables of the above optimization problem are the state vector of friendly resources for the whole optimization horizon, $\mathbf{X}^f$, and the corresponding control vector, $\mathbf{U}^f$. The state vector of enemy resources for the whole optimization horizon, $\mathbf{X}^e$, is not known. But as it has already been mentioned in Section 3.3.1, we can assume that enemy resources evolve according to an uncertain or stochastic feedback law given by (3.26).

This optimization problem can be written in a matrix form as

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{X}^f - \mathbf{X}^e\|_{\ell_1} \\ \text{subject to} \quad & \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}^f_c - \mathbf{T}^f_{xu,c} \end{bmatrix} \cdot \begin{pmatrix} \mathbf{X}^f \\ \mathbf{U}^f \end{pmatrix} \leq \begin{bmatrix} \mathbf{T}^f_{xx,\mathrm{id}} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}^f_{xx_0,c} \end{bmatrix} \cdot \begin{pmatrix} \mathbf{x}^f_{\max} \\ \mathbf{x}^f[0] \end{pmatrix} \\ & \begin{bmatrix} \mathbf{I} & -\mathbf{T}^f_{xu} \end{bmatrix} \cdot \begin{pmatrix} \mathbf{X}^f \\ \mathbf{U}^f \end{pmatrix} = \begin{bmatrix} \mathbf{O} & \mathbf{T}^f_{xx_0} \end{bmatrix} \cdot \begin{pmatrix} \mathbf{x}^f_{\max} \\ \mathbf{x}^f[0] \end{pmatrix} \\ \text{variables} \quad & \mathbf{X}^f \in \Re_+^{N_p n_x}, \quad \mathbf{U}^f \in \Re_+^{N_p n_u} \end{aligned} \tag{3.39}$$

The above optimization problem is a convex optimization problem, since the objective function is convex (ℓ_1 norm) and the constraints are linear. This problem can be converted to an *equivalent linear optimization problem.*

3.4.6 Equivalent linear optimization problem

The objective function of the convex optimization problem given by equation (3.39) can be written equivalently as

$$\|\mathbf{X}^f - \mathbf{X}^e\|_{\ell_1} = \sum_{t=1}^{N_p} \left\{ \sum_{i=1}^{n_s} \left\{ \sum_{j=1}^{n_r} \left| x^f_{s_i,r_j}[t] - x^e_{s_i,r_j}[t] \right| \right\} \right\} \tag{3.40}$$

This cost is always finite, since the state vectors of both friendly and enemy resources are non-negative and bounded from above. Thus, this cost is bounded from above by a positive number. We observe that $\left| x^f_{s_i,r_j}[t] - x^e_{s_i,r_j}[t] \right|$ is the smallest number $y_{s_i,r_j}[t] \in \Re_+$ that satisfies

$$\left(x^f_{s_i,r_j}[t] - x^e_{s_i,r_j}[t] \right) \leq y_{s_i,r_j}[t] \quad \text{and} \quad -\left(x^f_{s_i,r_j}[t] - x^e_{s_i,r_j}[t] \right) \leq y_{s_i,r_j}[t] \tag{3.41}$$

$$\forall t \in \{1, 2, \ldots, N_p\}, \quad \forall i \in \{1, 2, \ldots, n_s\}, \quad \forall j \in \{1, 2, \ldots, n_r\}$$

Instead of minimizing the sum of piecewise linear convex functions of (3.40), we can minimize the sum of their upper bounds, which is a linear function. Since this new objective function is linear (convex), the convexity of the problem is preserved, and the new optimization problem is equivalent with the initial one.

The new objective function is

$$\text{minimize} \quad \mathbf{1}^T \cdot \mathbf{Y} \tag{3.42}$$

where

$$\mathbf{1} = \begin{pmatrix} 1 \\ 1 \\ \ldots \\ 1 \end{pmatrix} \in \Re^{N_p n_x}, \quad \mathbf{Y} = \begin{pmatrix} \mathbf{y}[1] \\ \mathbf{y}[2] \\ \ldots \\ \mathbf{y}[N_p] \end{pmatrix} \in \Re_+^{N_p n_x}$$

and

$$\mathbf{y}[t] = \left(\begin{array}{cccccccc} y_{s_1,r_1} & y_{s_1,r_2} & \cdots & y_{s_1,r_{n_r}} & \cdots & y_{s_{n_s},r_1} & y_{s_{n_s},r_2} & \cdots & y_{s_{n_s},r_{n_r}} \end{array} \right)^T [t] \in \Re_+^{n_x}$$

$$\forall t \in \{1, 2, \ldots, N_p\}$$

while the new linear constraints of equation (3.41) can be written as

$$\left(\mathbf{X}^f - \mathbf{X}^e\right) \leq \mathbf{Y} \quad \text{and} \quad -\left(\mathbf{X}^f - \mathbf{X}^e\right) \leq \mathbf{Y} \tag{3.43}$$

Thus, the convex optimization problem of (3.39) is equivalent to the following linear optimization problem:

$$\begin{aligned} \text{minimize} \quad & \mathbf{1}^T \cdot \mathbf{Y} \\ \text{subject to} \quad & \left[\begin{array}{cc} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_c^f - \mathbf{T}_{xu,c}^f \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{X}^f \\ \mathbf{U}^f \end{array} \right) \leq \left[\begin{array}{cc} \mathbf{T}_{xx,\text{id}}^f & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_{xx_0,c}^f \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{x}_{\max}^f \\ \mathbf{x}^f[0] \end{array} \right) \\ & \left(\mathbf{X}^f - \mathbf{X}^e\right) \leq \mathbf{Y}, \quad -\left(\mathbf{X}^f - \mathbf{X}^e\right) \leq \mathbf{Y} \\ & \left[\begin{array}{cc} \mathbf{I} & -\mathbf{T}_{xu}^f \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{X}^f \\ \mathbf{U}^f \end{array} \right) = \left[\begin{array}{cc} \mathbf{O} & \mathbf{T}_{xx_0}^f \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{x}_{\max}^f \\ \mathbf{x}^f[0] \end{array} \right) \\ \text{variables} \quad & \mathbf{X}^f \in \Re_+^{N_p n_x}, \quad \mathbf{U}^f \in \Re_+^{N_p n_u}, \quad \mathbf{Y} \in \Re_+^{N_p n_x} \end{aligned} \tag{3.44}$$

which can be written equivalently as

$$\begin{aligned} \min \quad & \left[\begin{array}{ccc} \mathbf{O} & \mathbf{O} & \mathbf{1}^T \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{X}^f \\ \mathbf{U}^f \\ \mathbf{Y} \end{array} \right) \\ \text{s.t.} \quad & \left[\begin{array}{ccc} \mathbf{I} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_c^f - \mathbf{T}_{xu,c}^f & \mathbf{O} \\ \mathbf{I} & \mathbf{O} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{O} & -\mathbf{I} \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{X}^f \\ \mathbf{U}^f \\ \mathbf{Y} \end{array} \right) \leq \left[\begin{array}{ccc} \mathbf{T}_{xx,\text{id}}^f & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_{xx_0,c}^f & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{I} \\ \mathbf{O} & \mathbf{O} & -\mathbf{I} \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{x}_{\max}^f \\ \mathbf{x}^f[0] \\ \mathbf{X}^e \end{array} \right) \\ & \left[\begin{array}{ccc} \mathbf{I} & -\mathbf{T}_{xu}^f & \mathbf{O} \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{X}^f \\ \mathbf{U}^f \\ \mathbf{Y} \end{array} \right) = \left[\begin{array}{ccc} \mathbf{O} & \mathbf{T}_{xx_0}^f & \mathbf{O} \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{x}_{\max}^f \\ \mathbf{x}^f[0] \\ \mathbf{X}^e \end{array} \right) \\ \text{variables} \quad & \mathbf{X}^f \in \Re_+^{N_p n_x}, \quad \mathbf{U}^f \in \Re_+^{N_p n_u}, \quad \mathbf{Y} \in \Re_+^{N_p n_x} \end{aligned} \tag{3.45}$$

We state again that the state vector of the enemy resources for the whole optimization horizon, $\mathbf{X}^e$, is not known, and that enemy resources evolve according to an assumed uncertain or stochastic feedback law, represented by (3.26). According to this equation, the linear constraints of (3.43) are written as

$$\begin{aligned} \left(\mathbf{X}^f - \mathbf{T}^e_{xx_0,G} \cdot \mathbf{x}^e[0]\right) &\leq \mathbf{Y} \\ -\left(\mathbf{X}^f - \mathbf{T}^e_{xx_0,G} \cdot \mathbf{x}^e[0]\right) &\leq \mathbf{Y} \end{aligned} \tag{3.46}$$

Finally, the linear optimization problem of (3.45) is equivalent to

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T \cdot \mathbf{z} \\ \text{subject to} \quad & \mathbf{D} \cdot \mathbf{z} - \mathbf{E} \cdot \mathbf{d} \leq 0 \\ & \mathbf{F} \cdot \mathbf{z} - \mathbf{H} \cdot \mathbf{d} = 0 \\ \text{variables} \quad & \mathbf{z} \in \Re_+^{N_p(2n_x+n_u)} \end{aligned} \tag{3.47}$$

where

$$\mathbf{c} = \begin{pmatrix} \mathbf{O} \\ \mathbf{O} \\ \mathbf{1}^T \end{pmatrix}, \quad \mathbf{z} = \begin{pmatrix} \mathbf{X}^f \\ \mathbf{U}^f \\ \mathbf{Y} \end{pmatrix}, \quad \mathbf{d} = \begin{pmatrix} \mathbf{x}^f_{\max} \\ \mathbf{x}^f[0] \\ \mathbf{x}^e[0] \end{pmatrix}$$

$$\mathbf{D} = \begin{bmatrix} \mathbf{I} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}^f_c - \mathbf{T}^f_{xu,c} & \mathbf{O} \\ \mathbf{I} & \mathbf{O} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{O} & -\mathbf{I} \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} \mathbf{T}^f_{xx,\mathrm{id}} & \mathbf{O} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}^f_{xx_0,c} & \mathbf{O} \\ \mathbf{O} & \mathbf{O} & \mathbf{T}^e_{xx_0,G} \\ \mathbf{O} & \mathbf{O} & -\mathbf{T}^e_{xx_0,G} \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} \mathbf{I} & -\mathbf{T}^f_{xu} & \mathbf{O} \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{O} & \mathbf{T}^f_{xx_0} & \mathbf{O} \end{bmatrix}$$

which is a linear optimization problem in standard form.

In other words, by removing attrition losses from the original model of friendly and enemy resources flow, and by introducing a stochastic feedback matrix for the enemy resources, we finally obtain a linear programming optimization problem.

3.4.7 Alternative objective function

The previously described linear programming optimization problem of (3.47) is equivalent to the convex programming optimization problem of (3.39). However, in order to avoid converting the initial convex objective function to an equivalent linear one and introducing new (slack) variables, it is preferable to select a linear objective function from the start. In particular, one linear objective function could be

$$\text{minimize} \quad -\left(\mathbf{X}^e\right)^T \cdot \mathbf{X}^f \tag{3.48}$$

which is an affine (linear) function of the state vector of friendly resources. In other words, the minimum value of this function is attained when the maximum possible interceptions of enemy resources by friendly resources occur in the arena of sectors and within the next N_p stages.

This objective function, accompanied with the dynamic constraints of (3.25), the state constraints of (3.27) and the control constraints of (3.29) and (3.34), form the following linear programming optimization problem for friendly planning:

$$\begin{aligned} \text{minimize} \quad & -\left(\mathbf{X}^e\right)^T \cdot \mathbf{X}^f \\ \text{subject to} \quad & \mathbf{X}^f = \mathbf{T}^f_{xx_0} \cdot \mathbf{x}^f[0] + \mathbf{T}^f_{xu} \cdot \mathbf{U}^f \\ & \mathbf{0} \leq \mathbf{X}^f \leq \mathbf{X}^f_{\max} \\ & \mathbf{0} \leq \mathbf{U}^f \\ & \left(\mathbf{T}_c - \mathbf{T}^f_{xu,c}\right) \cdot \mathbf{U}^f \leq \mathbf{T}_{xx_0,c} \cdot \mathbf{x}^f[0] \end{aligned} \tag{3.49}$$

Since the state vectors of both teams, $\mathbf{X}^f$ and $\mathbf{X}^e$, are always positive, the minimum of their inner product is attained when the size of their difference is minimized, i.e., when $\left\|\mathbf{X}^f - \mathbf{X}^e\right\|_{\ell_1}$ attains its minimum. In other words, the linear programming optimization program of (3.49) is equivalent to (i.e., has the same optimal solution as) the convex optimization problem of (3.39).

Equation (3.49) is written equivalently as

$$
\begin{aligned}
\text{minimize} \quad & -\left(\mathbf{X}^e\right)^T \cdot \mathbf{X}^f \\
\text{subject to} \quad & \left[\begin{array}{cc} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_c^f - \mathbf{T}_{xu,c}^f \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{X}^f \\ \mathbf{U}^f \end{array}\right) \leq \left[\begin{array}{cc} \mathbf{T}_{xx,\mathrm{id}}^f & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_{xx_0,c}^f \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{x}_{\max}^f \\ \mathbf{x}^f[0] \end{array}\right) \\
& \left[\begin{array}{cc} \mathbf{I} & -\mathbf{T}_{xu}^f \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{X}^f \\ \mathbf{U}^f \end{array}\right) = \left[\begin{array}{cc} \mathbf{O} & \mathbf{T}_{xx_0}^f \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{x}_{\max}^f \\ \mathbf{x}^f[0] \end{array}\right) \\
\text{variables} \quad & \mathbf{X}^f \in \Re_+^{N_p n_x}, \quad \mathbf{U}^f \in \Re_+^{N_p n_u}
\end{aligned}
\tag{3.50}
$$

According to the model simplifications of Section 3.3.1, we assume that the enemy resources follow an uncertain or stochastic feedback matrix, $\mathbf{G}^e$. In this case, state vector $\mathbf{X}^e$ is given by

$$
\mathbf{X}_{1\to N_p}^e = \mathbf{T}_{xx_0,G}^e \cdot \mathbf{x}^e[0] \tag{3.51}
$$

where $\mathbf{T}_{xx_0,G}^e$ was defined by (3.26).

Hence, the linear programming optimization problem of (3.49) is equivalent to

$$
\begin{aligned}
\text{minimize} \quad & -\left[\begin{array}{cc} \left(\mathbf{T}_{xx_0,G} \cdot \mathbf{x}^e[0]\right)^T & \mathbf{0}^T \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{X}^f \\ \mathbf{U}^f \end{array}\right) \\
\text{subject to} \quad & \left[\begin{array}{cc} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_c^f - \mathbf{T}_{xu,c}^f \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{X}^f \\ \mathbf{U}^f \end{array}\right) \leq \left[\begin{array}{cc} \mathbf{T}_{xx,\mathrm{id}}^f & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_{xx_0,c}^f \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{x}_{\max}^f \\ \mathbf{x}^f[0] \end{array}\right) \\
& \left[\begin{array}{cc} \mathbf{I} & -\mathbf{T}_{xu}^f \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{X}^f \\ \mathbf{U}^f \end{array}\right) = \left[\begin{array}{cc} \mathbf{O} & \mathbf{T}_{xx_0}^f \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{x}_{\max}^f \\ \mathbf{x}^f[0] \end{array}\right) \\
\text{variables} \quad & \mathbf{X}^f \in \Re_+^{N_p n_x}, \quad \mathbf{U}^f \in \Re_+^{N_p n_u}
\end{aligned}
\tag{3.52}
$$

If we compare the linear programming optimization problem of (3.47) with the previous one in (3.52), it is easily seen that the latter has $N_p n_x$ fewer variables, and $2N_p n_x$ fewer inequality constraints than the former. Therefore, the latter optimization problem is computationally more efficient, since the computational complexity of a linear problem grows (polynomially) in the number of variables and constraints.

3.4.8 Feasibility of the optimization problem

The question that arises now is whether the linear programming optimization problem of either (3.47) or (3.52) is feasible or not, i.e., whether there is a feasible solution. First of all, the feasible set of these optimization problems is not empty since there is at least one state vector that satisfies the positivity constraints, e.g. the initial state vector. Thus, there is at least one feasible solution.

The existence of feasible solutions does not necessarily guarantee the existence of an optimal (or basic) feasible solution, since the optimal cost could be infinite. However, the polyhedron of the constraints of (3.47) or (3.52) is bounded, since the state vector is bounded from above and below. Hence, the linear programming optimization problem has at least one optimal solution.

3.5 Receding horizon philosophy

Such an optimal feasible solution represents an optimal path for friendly resources (for N_p stages) so that the maximum possible enemy resources levels to be intercepted. However, both optimizations (3.47) and (3.52) do not contain the possible attrition losses of both teams, and the future positions of enemy resources are computed by using an assumed feedback control law. Thus, the use of a *receding horizon philosophy* is necessary.

Recall that the previously described linear optimization problems were the result of two model simplifications given in Section 3.3.1. These two model simplifications were the removal of attrition function from the state-space equations and the introduction of an assumed uncertain or stochastic state-feedback matrix for the enemy resources.

In other words, the model used for prediction and optimization is not the same as the "*plant*" to be controlled. In particular, the plant is described by the following

state-space equation

$$\Sigma_2 : \quad \begin{cases} \begin{pmatrix} \mathbf{x}^f \\ \mathbf{x}^e \end{pmatrix}^+ = \begin{pmatrix} \mathbf{x}^f \\ \mathbf{x}^e \end{pmatrix} + \begin{pmatrix} \mathbf{B}^f \\ \mathbf{O} \end{pmatrix} \cdot \mathbf{u}^f + \begin{pmatrix} \mathbf{O} \\ \mathbf{B}^e \end{pmatrix} \cdot \mathbf{u}^e - \begin{pmatrix} \mathbf{l}^f \\ \mathbf{l}^e \end{pmatrix} \\ \\ \mathbf{x}^f, \mathbf{x}^e \in \mathcal{P}, \quad \mathbf{u}^f \in \mathcal{C}(\mathbf{x}^f), \quad \mathbf{u}^e \in \mathcal{C}(\mathbf{x}^e) \end{cases} \tag{3.53}$$

while the model used for prediction and optimization was

$$\Sigma_2' : \quad \begin{cases} \begin{pmatrix} \mathbf{x}^f \\ \mathbf{x}^e \end{pmatrix}^+ = \begin{pmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{I} + \mathbf{B}^e\mathbf{G}^e \end{pmatrix} \cdot \begin{pmatrix} \mathbf{x}^f \\ \mathbf{x}^e \end{pmatrix} + \begin{pmatrix} \mathbf{B}^f \\ \mathbf{O} \end{pmatrix} \cdot \mathbf{u}^f \\ \\ \mathbf{x}^f, \mathbf{x}^e \in \mathcal{P}, \quad \mathbf{u}^f \in \mathcal{C}(\mathbf{x}^f), \quad \mathbf{G}^e\mathbf{x}^e \in \mathcal{C}(\mathbf{x}^e) \end{cases} \tag{3.54}$$

Because of the differences between these two models, we should expect significant discrepancies between the response of the plant and the response of the model used for prediction and optimization. For this reason, the result of the linear optimization problem of (3.47) or (3.52) is implemented according to a *receding horizon philosophy* described by Bemporad and Morari in [2].

According to this strategy, at time t only the first optimal input is applied to the plant. The remaining optimal inputs are not implemented. At time $t+1$ we measure the current states of both teams. These measurements include actual trajectories and attrition losses that occurred after the implementation of the first optimal control input. Using these measurements, a new optimal control problem is solved at time $t+1$.

In other words, the following algorithm is implemented. At time t:

1. Measure the new state vectors of both teams, $\mathbf{x}^f[t]$ and $\mathbf{x}^e[t]$.

2. Solve the linear programming (LP) optimization problem of (3.47) or (3.52). Let

$$(\mathbf{U}^*)^f = \left[\begin{array}{cccc} (\mathbf{u}^*)^f[t+0] & (\mathbf{u}^*)^f[t+1] & \cdots & (\mathbf{u}^*)^f[t+N_p] \end{array} \right]$$

be the optimal control for the whole optimization horizon, N_p.

3. Apply only the first optimal control, i.e., $\mathbf{u}^f[t] = (\mathbf{u}^*)^f\,[t+0]$.

4. Advance $t \leftarrow t+1$ and repeat.

With $\mathbf{u}^e[t]$ as the unknown control of enemy resources at time t, which is the disturbance of system Σ_2, the *receding horizon strategy* of friendly resources is described by the block diagram in Figure 3.3.

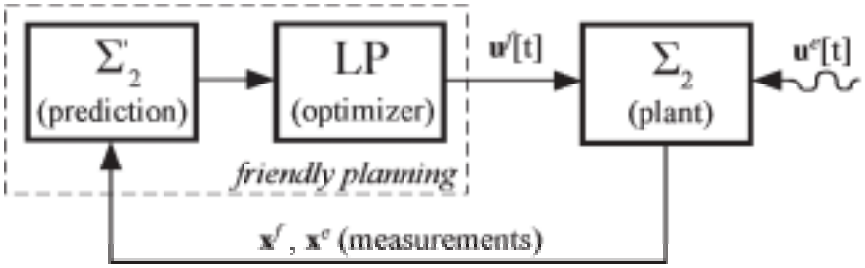


Figure 3.3: Basic structure of Receding Horizon Philosophy.

In a general case, the type and the size of the disturbances are generally unknown. However, in case of system Σ_2, the disturbances are the enemy resources, which are always state dependent and subject to positivity constraints similar to those of friendly resources. For this reason, the *receding horizon strategy* is always "*stabilizing*" in the sense that the state of friendly resources cannot be driven out of the pre-specified constraints. Hence, stability of the system in Figure 3.3 is always satisfied.

However, the result of this control strategy is not necessarily satisfactory. In other words, it is not guaranteed that friendly resources will cause the maximum possible attrition to enemy resources. That is because the linear programming optimization problem depends on the assumed feedback matrix $\mathbf{G}^e$, which does not necessarily predict accurately the future states of enemy resources. Thus, the computation of this feedback matrix is of vital importance in terms of the effectiveness of friendly planning.

3.6 Remarks

In this chapter, we examined the case where two different teams (friendly or enemy) of resources (of same types) evolve in the same arena of sectors. Assuming that these resources are subject to attrition losses when resources of different teams lie in the same sectors, we desire friendly optimal tactics in order for enemy's attrition to be maximized.

Although enemy resources can be viewed as a disturbance to the system of friendly resources, they are state dependent and subject to the same constraints as the friendly ones. Based on such constraints we derived a linear-programming-based planning for the friendly resources, accompanied with a receding horizon implementation, that optimizes the friendly paths.

The question that arises now is whether this planning for resource allocation can be used in deriving satisfactory paths for multi-vehicle systems. In the following chapter, we explore the effectiveness of the previously described optimization planning to multi-vehicle tasks.

CHAPTER 4

Multi-Vehicle Path Planning

4.1 Introduction

In this chapter we explore the utility of the linear-programming-based planning for resource allocation, described in the previous chapter, in deriving optimal paths for multi-vehicle control systems with adversaries. In particular, we consider the RoboFlag competition which involves two teams of robots with opposing interests, the defenders and the attackers. We derive control algorithms for both defense and attack that are based on the linear-programming-based planning for resource allocation. In this case, both defenders and attackers are modelled as different resource types in an arena of sectors. Moreover, since adversaries are modelled as state-dependent, various stochastic feedback laws based on their current position and/or velocity could describe their possible future attitude, which is included in the optimization.

4.2 The RoboFlag competition

The RoboFlag competition, described by D'Andrea and Murray in [12], involves two teams of robots. The goal of each team is to infiltrate the protected zone of the other team, get their flag, and bring it to their home base. However, in parallel, each team has to thwart its opponents from capturing its own flag. This game is quite similar to the well-known games "*capture the flag*" and "*paint-ball*". In addition, each team has to worry about some other parameters. For example, there are some parts of

the field that are off-limits, moving obstacles must be avoided, and each robot has a limited amount of fuel.

This problem seems quite complicated, and for this reason several simplified versions of this competition can be considered. One of them is the "RoboFlag Drill" introduced by Earl and D'Andrea in [16]. This simplified version involves again two teams of robots, the attackers and the defenders, on a playing field with a region at its center called the defense zone, as in Figure 4.1. In reference [16] the attackers are drones directed toward the center of the defense zone along a straight-line path at constant velocity which is known to the defenders. The objective of the defenders is to thwart the attackers from entering the defense zone by intercepting each attacker before it enters the zone. Once an attacker enters the defense zone or is intercepted by a defender it remains stationary for the remainder of the game. The objective of the "RoboFlag Drill" is to minimize the number of the attackers that enter the defense zone over the duration of the drill.

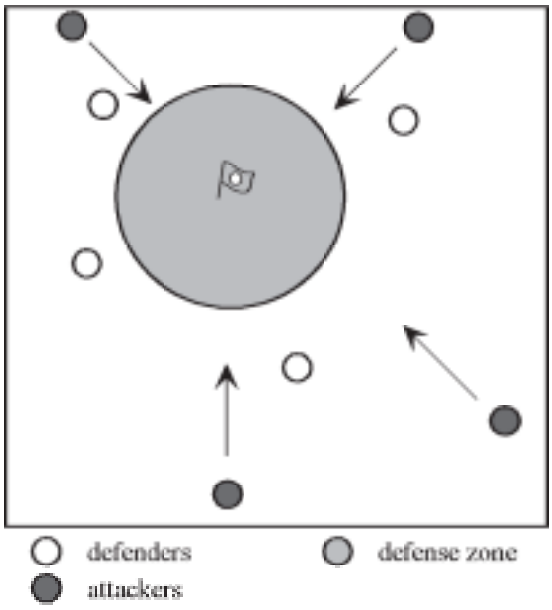


Figure 4.1: A simplified version of the RoboFlag competition, *Roboflag Drill*.

This simplified version of the Roboflag competition leads to several other generilations. In particular, the attackers could follow random paths, which are unknown to the defenders, towards the defense zone. An even more complicated version than the previous one could involve efficient attacking tactics, where the attackers decide their next positions based on their relative distance from the defense zone and the defenders. In both these two versions of the game, the objective of the defenders is again the interception of the maximum possible number of attackers.

In the following section, we explore the utility of a linear-programming-based planning for deriving tactics for both the defenders and the attackers. These methods are similar to those derived in Chapter 3 for friendly resource allocation planning.

4.3 Linear-programming-based defense planning

We pursue an approach that allows the use of linear programming in conjunction with a receding horizon philosophy in order to find optimal paths for the defenders. To this end, both defenders and attackers will be considered as two different teams of resources. In particular, defenders will be considered as friendly resources, while attackers will be considered as enemy resources. Each unit of the defenders or the attackers could be viewed as a different resource type of the team. However, since each attacker is subject to "attrition losses" (i.e., it becomes inactive after being intercepted by a defender), we assume that each resource type (unit) of the attackers becomes inactive when it is intercepted by any resource type (unit) of the defenders.

The playing ground is divided into sectors, and the state for this system (resources flow) is defined as the level of a resource type at each sector. Moreover, in order for a resource level to correspond to a unit of the defenders or the attackers, we assign resource level "1" to the sector in which a unit lies, and "0" to the sector at which there is no unit. In other words, the resource level can take only two integer values, "0" and

"1". Moreover, defenders are not allowed to enter their defense zone, which means that the sectors of the defense zone cannot be reached by the defenders' resources.

Summarizing, we assume that:

- Defenders and attackers are considered as friendly and enemy resources, respectively.
- Each defender and each attacker could be viewed as a different resource type.
- Any resource type (unit) of the attackers can be intercepted by any resource type (unit) of the defenders.
- The playing ground is divided into sectors, and the level of any resource type at any sector takes on only two integer values, "0" or "1".
- The defenders' resources cannot reach the sectors of their defense zone.

Under these assumptions, Figure 4.1 takes the form of Figure 4.2, where the position of each unit of both teams corresponds to a sector. In addition, although only the attackers are subject to attrition losses, this problem can be transformed to a linear-programming-based planning in a similar manner to the described in Chapter 3. According to that procedure, we solve a linear programming optimization problem for friendly planning that maximizes the number of interceptions between friendly and enemy resources within N_p time stages. Since enemy resources are subject to attrition losses, we implement only the first optimal control. At the next stage, we measure the new states of both teams, and we solve again the linear optimization problem.

4.3.1 Binary constraints

Our final goal is the derivation of a linear programming optimization problem for defense which will be similar to those of (3.47) and (3.52). However, according to

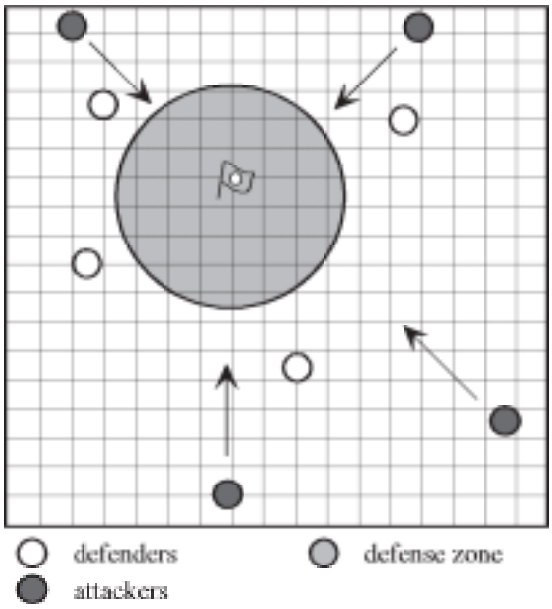


Figure 4.2: The playing area is separated into sectors.

the previous assumptions, both friendly and enemy resource levels take on only two integer values, "0" and "1" (*binary constraints*). In other words, a new constraint must be added to the linear optimization problem. Let the superscripts "d" and "a" denote defenders and attackers, respectively. For the defenders, this binary constraint is equivalently written as

$$x^d_{s_i,r_j} = 0 \quad \text{or} \quad x^d_{s_i,r_j} = 1, \quad \forall s_i \in \mathcal{S}, \quad \forall r_j \in \mathcal{R}^d \tag{4.1}$$

where $\mathcal{R}^d$ is the set of defenders' resource types. This constraint can be written equivalently as

$$\mathbf{x}^d \leq \mathbf{x}_{\max} = \mathbf{1}, \quad \mathbf{x}^d \in \mathcal{Z}_+^{n_x^d} \tag{4.2}$$

where $\mathcal{Z}_+$ is the set of non-negative integer numbers, $n_x^d = n_s n_r^d$ is the number of defenders' states, and n_r^d is the number of defenders' resource types, i.e., the number of defenders' units.

Similarly, if $\mathbf{x}^a$ is the attackers' state vector, then

$$\mathbf{x}^a \leq \mathbf{x}_{\max} = \mathbf{1}, \quad \mathbf{x}^a \in \mathcal{Z}_+^{n_x^a} \tag{4.3}$$

where $n_x^a = n_s n_r^a$ is the number of attackers' states, and n_r^a is the number of attackers' resource types, i.e., the number of attackers' units.

4.3.2 Defense-zone constraints

The defenders' resources are not allowed to enter the defense zone (*defense-zone constraints*). We would prefer to express this constraint as a linear function of the defenders' state vector. To this end, we denote $\mathcal{S}^{dz}$ as the set of sectors that belong to the defense zone and we define the vector $\mathbf{x}_1^{dz}$ such that,

$$\mathbf{x}_1^{dz}(i,1) = \begin{cases} 1, & \text{if } \; s_i \in \mathcal{S}^{dz} \\ \\ 0, & \text{if } \; s_i \notin \mathcal{S}^{dz} \end{cases} \in \mathcal{Z}_+^{n_s}, \quad i \in \{1,2,3,\ldots,n_s\} \tag{4.4}$$

In other words, $\mathbf{x}_1^{dz}$ represents the state vector of one resource type that lies in the defense zone. The subscript "1" corresponds to the number of resource types.

We also define the vector

$$\mathbf{x}^{dz} = \left[\begin{array}{cccc} \left(\mathbf{x}_1^{dz}\right)^T & \left(\mathbf{x}_1^{dz}\right)^T & \ldots & \left(\mathbf{x}_1^{dz}\right)^T \end{array} \right]^T \in \mathcal{Z}_+^{n_x^d} \tag{4.5}$$

which represents the state vector of n_r^d different resource types that lie in the defense zone.

In this case, the *defense-zone constraints* can be written as

$$\left(\mathbf{x}^{dz}\right)^T \cdot \mathbf{x}^d = 0 \tag{4.6}$$

There is no constraint of this type for the attackers since they are allowed to enter the defense zone.

4.3.3 Mixed integer programming optimization problem

Since, the *binary* and *defense-zone constraints* of (4.2), (4.3) and (4.6) are linear, we can add them to one of the linear programming optimization problems derived

in Chapter 3, see equations (3.47) and (3.52). The resulting optimization is a linear programming optimization problem that can be used in deriving optimal paths for the defenders.

As explained in Section 3.4.7, the linear programming problem of (3.52) is computationally more efficient than that of (3.47). For this reason, and since they are equivalent, only the former one will be used in deriving optimal paths for the defenders. However, these linear programming optimization problems were derived for the case where the number of friendly resource types is equal to the number of enemy resource types. Thus, assuming that the number of defenders' resources is equal to the number of attackers' resources, i.e.,

$$n_r^d = n_r^a \tag{4.7}$$

the linear programming optimization problem of (3.52) augmented with the *binary* and *defense-zone constraints* of (4.2), (4.3) and (4.6) formulate a *mixed integer programming optimization problem*, that is

$$\begin{aligned}
&\text{minimize} \quad -\left[\begin{array}{cc} \left(\mathbf{X}^a\right)^T & \mathbf{0}^T \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{X}^d \\ \mathbf{U}^d \end{array}\right) \\
&\text{subject to} \quad \left[\begin{array}{cc} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_c^d - \mathbf{T}_{xu,c}^d \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{X}^d \\ \mathbf{U}^d \end{array}\right) \leq \left[\begin{array}{cc} \mathbf{T}_{xx,\mathrm{id}}^d & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_{xx_0,c}^d \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{1} \\ \mathbf{x}^d[0] \end{array}\right) \\
&\qquad\qquad \left[\begin{array}{cc} \mathbf{I} & -\mathbf{T}_{xu}^d \\ \left(\mathbf{X}^{dz}\right)^T & \mathbf{O} \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{X}^d \\ \mathbf{U}^d \end{array}\right) = \left[\begin{array}{cc} \mathbf{O} & \mathbf{T}_{xx_0}^d \\ \mathbf{O} & \mathbf{O} \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{1} \\ \mathbf{x}^d[0] \end{array}\right) \\
&\text{variables} \quad \mathbf{X}^d \in \mathcal{Z}_+^{N_p n_x^d}, \quad \mathbf{U}^d \in \Re_+^{N_p n_u^d}
\end{aligned} \tag{4.8}$$

where superscripts "d" and "a" correspond to "defenders" and "attackers", respectively, and $\mathbf{X}^{dz}$ is defined according to the definitions of $\mathbf{X}^d$, $\mathbf{X}^a$, i.e.,

$$\mathbf{X}^{dz} = \mathbf{X}_{1\rightarrow N_p}^{dz} = \left(\begin{array}{c} \mathbf{x}^{dz}[1] \\ \mathbf{x}^{dz}[2] \\ \cdots \\ \mathbf{x}^{dz}[N_p] \end{array}\right) = \left(\begin{array}{c} \mathbf{x}^{dz} \\ \mathbf{x}^{dz} \\ \cdots \\ \mathbf{x}^{dz} \end{array}\right) \in \mathcal{Z}_+^{N_p n_x^d} \tag{4.9}$$

Although there are several methods that can be used for computing the optimal solution of a mixed integer optimization problem, such as *cutting plane methods* or *branch and bound* [4], we prefer to solve a linear programming problem instead. The main reason for this preference is the computational complexity of an integer problem.

Moreover, given the availability of several efficient algorithms for linear programming, the choice of a formulation, although important, does not critically affect our ability to solve the problem. This is not always the case for integer programming formulations, where, especially in large optimization problems the choice of a formulation is crucial.

In order to avoid these handicaps of integer programming, we transform the *mixed integer problem* of (4.8) into a *linear-programming-based optimization planning*. This planning includes the following steps:

1. We introduce the *linear programming relaxation* of the *mixed integer programming problem* of (4.8).

2. Given the non-integer optimal solution of the *linear programming relaxation*, we compute a suboptimal solution for the *mixed integer programming problem*.

3. We apply this solution according to a *receding horizon philosophy*.

4.3.4 Linear programming relaxation

The *linear programming relaxation* of the *mixed-integer programming problem* of (4.8) is

$$
\begin{aligned}
\text{minimize} \quad & -\left[\begin{array}{cc} (\mathbf{X}^a)^T & \mathbf{0}^T \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{X}^d \\ \mathbf{U}^d \end{array}\right) \\
\text{subject to} \quad & \left[\begin{array}{cc} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_c^d - \mathbf{T}_{xu,c}^d \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{X}^d \\ \mathbf{U}^d \end{array}\right) \leq \left[\begin{array}{cc} \mathbf{T}_{xx,\mathrm{id}}^d & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_{xx_0,c}^d \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{1} \\ \mathbf{x}^d[0] \end{array}\right) \\
& \left[\begin{array}{cc} \mathbf{I} & -\mathbf{T}_{xu}^d \\ \left(\mathbf{X}^{dz}\right)^T & \mathbf{O} \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{X}^d \\ \mathbf{U}^d \end{array}\right) = \left[\begin{array}{cc} \mathbf{O} & \mathbf{T}_{xx_0}^d \\ \mathbf{O} & \mathbf{O} \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{1} \\ \mathbf{x}^d[0] \end{array}\right) \\
\text{variables} \quad & \mathbf{X}^d \in \Re_+^{N_p n_x^d}, \quad \mathbf{U}^d \in \Re_+^{N_p n_u^d}
\end{aligned}
\tag{4.10}
$$

where now the state vector $\mathbf{X}^d$ can take any value between $\mathbf{0}$ and $\mathbf{1}$. For this reason, the polyhedron defined by the linear constraints of this relaxation problem includes the corresponding polyhedron of the mixed integer programming problem. More specifically, if P_{LP}, P_{MI} are these two polyhedrons, respectively, then both of them contain exactly the same set of integer solutions. Thus, if the problem were two dimensional, then these two polyhedrons would have the form of Figure 4.3.

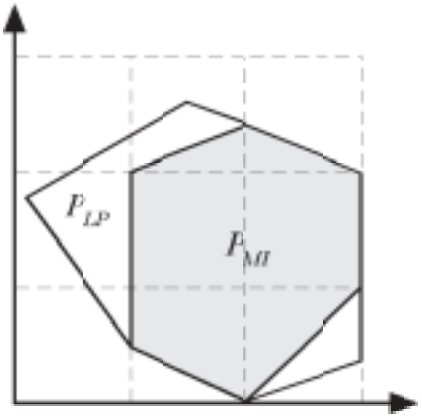


Figure 4.3: The two polyhedra P_{LP} and P_{MI} contain exactly the same set of integer solutions.

Therefore, if an optimal solution to the relaxation is feasible for the mixed inte-

ger programming problem, it is also an optimal solution to its linear programming relaxation.

4.3.5 Computation of a suboptimal solution

In general, the solution of the *linear programming relaxation* is a non-integer vector, which means that this solution does not belong to the feasible set of the *mixed integer programming problem*. However, as explained in Section 3.5, the optimal solution of the latter optimization problem is applied according to a *receding horizon philosophy*. Since this strategy implements only the first optimal control, we are only interested in the first optimal solution of the mixed integer programming problem.

Given the optimal control of the *linear programming relaxation* for the whole optimization horizon, N_p,

$$(\mathbf{U}^*)^d = \left[\begin{array}{cccc} (\mathbf{u}^*)^d[t+0] & (\mathbf{u}^*)^d[t+1] & \cdots & (\mathbf{u}^*)^d[t+N_p] \end{array} \right]$$

we are interested only in the first optimal control, i.e., $(\mathbf{u}^*)^d[t] = (\mathbf{u}^*)^d[t+0]$, which will generally be a non-integer vector between $\mathbf{0}$ and $\mathbf{1}$.

Given this first optimal solution of the *linear programming relaxation*, we construct a *suboptimal solution* of the *mixed integer programming problem*. In particular, among the resource levels that exit from or remains at a sector, we pick up the maximum of them, to which we assign the value "1", while the rest of them are assigned the value "0". In this way, we define an integer control input that belongs to the feasible set of the *mixed integer programming problem* of (4.8), while, in parallel, the sum of the resource levels remains the same as that of the previous stage.

Algorithmically, according to the definition of the function $\mathcal{U}_{\text{out}} : \mathcal{SN}(\mathcal{S}, \mathcal{R}) \times \mathcal{R} \times \mathcal{S} \rightarrow \{1, 2, \ldots, n_u\}$ (see equation (2.12) in page 18), we define the corresponding function of the defenders' resources as

$$\mathcal{U}^d_{\text{out}} : \mathcal{SN}^d(\mathcal{S}, \mathcal{R}^d) \times \mathcal{R}^d \times \mathcal{S} \rightarrow \left\{1, 2, \ldots, n^d_u\right\} \tag{4.11}$$

where $\mathcal{SN}^d(\mathcal{S}, \mathcal{R}^d)$ is the set of neighboring sectors of the defenders' resources. This function maps a triple of the form (s_k, r_j, s_i), where $s_i \in \mathcal{S}$, $r_j \in \mathcal{R}^d$ and $s_k \in \mathcal{SN}^d(s_i, r_j)$, to the row number of the control vector, $\mathbf{u}^d$, that corresponds to the control level $u^d_{s_k \leftarrow s_i, r_j}$, i.e.,

$$\mathbf{u}^d \left(\mathcal{U}^d_{\text{out}} \left(s_k, r_j, s_i \right), 1 \right) = u^d_{s_k \leftarrow s_i, r_j} \tag{4.12}$$

Let $\tilde{\mathbf{u}}^d[t]$ be a suboptimal solution of the mixed integer programming problem at time t. This solution can be constructed in the following way: For each sector $s_i \in \mathcal{S}$ and each resource type $r_j \in \mathcal{R}^d$,

1. Define the set

$$\mathcal{UN}^d(s_i, r_j) = \left\{ \mathcal{U}^d_{\text{out}}(s_k, r_j, s_i) \;\; | \;\; s_k \in \mathcal{SN}^d(s_i, r_j) \right\}$$

that includes the row numbers of the control vector $\mathbf{u}^d$ which correspond to the control inputs $\left\{ u^d_{s_k \leftarrow s_i, r_j} \;\; | \;\; s_k \in \mathcal{SN}^d(s_i, r_j) \right\}$.

2. If $x^d_{s_i, r_j} \neq 0$, find the row number, θ^*, of $(\mathbf{u}^*)^d [t]$ that corresponds to the maximum control level among those ones that exit from sector s_i, i.e.,

$$\theta^* = \arg \max_{\theta \in \mathcal{UN}^d(s_i, r_j)} \left\{ (\mathbf{u}^*)^d [t](\theta, 1) \right\}$$

   (a) if $(\mathbf{u}^*)^d [t](\theta^*, 1) \geq 1 - \sum_{\theta \in \mathcal{UN}^d(s_i, r_j)} \left\{ (\mathbf{u}^*)^d [t](\theta, 1) \right\}$, which means that the maximum control level that exits from sector s_i is greater than the resource level that remains at that sector, set

$$\tilde{\mathbf{u}}^d[t](\theta, 1) = \begin{cases} 1 & , \quad \text{if } \theta = \theta^* \\ 0 & , \quad \text{if } \theta \neq \theta^* \end{cases} \quad , \quad \theta \in \mathcal{UN}^d(s_i, r_j)$$

   which implies that the resource level $x^d_{s_i, r_j}$ will move to the neighboring sector $s_k \in \mathcal{SN}^d(s_i, r_j)$ that satisfies $\mathcal{U}^d_{\text{out}}(s_k, r_j, s_i) = \theta^*$.

(b) otherwise, set

$$\tilde{\mathbf{u}}^d[t](\theta, 1) = 0, \quad \forall \theta \in \mathcal{UN}^d(s_i, r_j).$$

which implies that the resource level $x^d_{s_i,r_j}$ will remain at sector s_i.

3. If $x^d_{s_i,r_j} = 0$, set

$$\tilde{\mathbf{u}}^d[t](\theta, 1) = 0, \quad \forall \theta \in \mathcal{UN}^d(s_i, r_j).$$

The resulting integer control vector, $\tilde{\mathbf{u}}^d[t]$, belongs to the feasible set of the *mixed integer programming problem* of (4.8), while the sum of the resource levels remains the same as the one of the previous stage.

4.3.6 Control algorithm for defense

We computed a suboptimal solution of the *mixed integer programming problem*, (4.8), by solving only its *linear programming relaxation*, (4.10). This solution is implemented in a *receding horizon manner*. In particular, the linear-programming-based planning for defense contains the following steps:

1. Measure the new state vectors of the defenders, $\mathbf{x}^d$, and the attackers, $\mathbf{x}^a$.

2. Solve the *linear programming relaxation*, (4.10), of the *mixed integer programming problem*, (4.8). Let

$$(\mathbf{U}^*)^d = \left[\begin{array}{cccc} (\mathbf{u}^*)^d[t+0] & (\mathbf{u}^*)^d[t+1] & \cdots & (\mathbf{u}^*)^d[t+N_p] \end{array} \right]$$

be its optimal control solution for the whole optimization horizon, N_p.

3. Given the first optimal control, $(\mathbf{u}^*)^d[t] = (\mathbf{u}^*)^d[t+0]$ of the *linear programming relaxation*, compute a suboptimal solution, $\tilde{\mathbf{u}}^d[t]$, for the *mixed integer programming problem*, according to Section 4.3.5.

4. Apply only the suboptimal solution, $\tilde{\mathbf{u}}^d[t]$.

5. Advance $t \leftarrow t+1$ and repeat.

4.3.7 Computational simplifications

The previously described linear-programming-based path planning for the defenders can be used for intercepting a group of attackers that try to enter the defense zone. Both the mixed integer programming optimization problem of (4.8) and its linear programming relaxation of (4.10) were derived by assuming that each unit of the defenders and the attackers is considered as a different resource type. Moreover, we assumed for the sake of simplicity that the number of defenders' resource types (units) is equal to the number of attackers' resource types (units), i.e., $n_r^d = n_r^a$.

Even though the resulting optimization problem is linear, we would prefer a simpler formulation of the same problem with fewer variables. Furthermore, we prefer that the number of defenders' resource types (units) could be different than the number of attackers' resource types (units). To this end, we define a new state vector for each team, $\mathbf{x}_1^d$ and $\mathbf{x}_1^a$, where the units of each team are considered as one resource type. In this case, the dimension of both these two vectors does not depend on the number of units and is equal to the number of sectors, n_s. In this way, we reduce the number of variables of the optimization problem, while we can model situations where the number of defenders' units is different than the number of attackers' units. In other words,

$$\mathbf{x}_1^d \leq \mathbf{1}, \quad \mathbf{x}_1^a \leq \mathbf{1}, \quad \text{where} \quad \mathbf{x}_1^d, \mathbf{x}_1^a \in \mathcal{Z}_+^{n_s} \tag{4.13}$$

while the corresponding control vector of the defenders is

$$\mathbf{u}_1^d \leq \mathbf{1}, \quad \text{where} \quad \mathbf{u}_1^d \in \mathcal{Z}_+^{n_s} \tag{4.14}$$

Similarly, the state vector of the defense zone is defined as

$$\mathbf{x}_1^{dz} \leq \mathbf{1}, \quad \text{where} \quad \mathbf{x}_1^{dz} \in \mathcal{Z}_+^{n_s} \tag{4.15}$$

By using these new state and control vectors, the linear programming relaxation of the mixed integer programming problem of (4.8) can be written equivalently as

$$\begin{aligned}
\text{minimize} \quad & -\left[\begin{array}{cc} (\mathbf{X}_1^a)^T & \mathbf{0}^T \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{X}_1^d \\ \mathbf{U}_1^d \end{array}\right) \\
\text{subject to} \quad & \left[\begin{array}{cc} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_c^d - \mathbf{T}_{xu,c}^d \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{X}_1^d \\ \mathbf{U}_1^d \end{array}\right) \leq \left[\begin{array}{cc} \mathbf{T}_{xx,\mathrm{id}}^d & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_{xx_0,c}^d \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{1} \\ \mathbf{x}_1^d[0] \end{array}\right) \\
& \left[\begin{array}{cc} \mathbf{I} & -\mathbf{T}_{xu}^d \\ \left(\mathbf{X}_1^{dz}\right)^T & \mathbf{O} \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{X}_1^d \\ \mathbf{U}_1^d \end{array}\right) = \left[\begin{array}{cc} \mathbf{O} & \mathbf{T}_{xx_0}^d \\ \mathbf{O} & \mathbf{O} \end{array}\right] \cdot \left(\begin{array}{c} \mathbf{1} \\ \mathbf{x}_1^d[0] \end{array}\right) \\
\text{variables} \quad & \mathbf{X}_1^d \in \Re_+^{N_p n_x^d}, \quad \mathbf{U}_1^d \in \Re_+^{N_p n_u^d}
\end{aligned} \tag{4.16}$$

where the vectors $\mathbf{X}_1$ and $\mathbf{U}_1$ are defined according to the definition of the vectors $\mathbf{X}$ and $\mathbf{U}$, respectively.

Note that this linear programming problem has fewer number of variables than that of (4.10). In particular, the latter one has $N_p n_x^d = N_p n_s n_r^d$ state variables and $N_p n_u^d = N_p n_r^d n_s(n_s - 1)$ control variables, where the number of defenders' resource types n_r^d coincides with the number of defenders' units. On the other hand, the optimization problem of (4.16) has $N_p n_s$ state variables and $N_p n_s(n_s - 1)$ control variables. Thus, the number of variables was reduced by

$$N_p n_s \left(n_r^d - 1\right) + N_p \left(n_r^d - 1\right) n_s(n_s - 1) \quad = \quad N_p n_s^2 \left(n_r^d - 1\right)$$

4.3.8 Alternative objective functions

According to the linear-programming-based defense planning, defenders objective function is the maximization of the inner product of the defenders' state vector with the attackers' state vector. Thus, we should expect that the defenders will move towards the attackers, since their interception will increase their inner product.

However, this objective function does not include any information about the distance between the defenders and their defense zone. In other words, the defenders will move towards the attackers without taking into account their distance from the defense zone. On the contrary, we would rather the defenders move towards the attackers, but at the same time stay reasonably close to the defense zone. The advantages of such a policy are the following:

- The defender spends less energy, since does not follow an attacker that moves far away from the defense zone.
- Since defenders move about the defense zone and within a small distance, the possibility of intercepting an attacker that tries to enter the defense zone is greater.

Thus, such a defense policy must provide a measure of how important is for the defenders to move towards the attackers or to stay closer to the defense zone. Since the inner product $-\left(\mathbf{X}_1^a\right)^T \cdot \mathbf{X}_1^d$, which was used as an objective function in (4.16), represents the cost of staying away from the attackers, we are interested in finding a similar affine (linear) function of $\mathbf{X}_1^d$, that will represent the cost of staying away from the defense zone. In that case, a possible objective function could be a weighted sum of these two linear functions of $\mathbf{X}_1^d$.

To this end, we define a small zone about the defense zone as the "*low-energy zone*" of the defenders that is denoted as "lz" and it is shown in Figure 4.4. The defenders that remain within this zone spend less energy, because they take advantage of the fact that eventually attackers will move closer to the defense zone. If $\mathcal{S}^{lz}$ denotes the set of sectors that belong to the low-energy zone, we define the following state vector:

$$\mathbf{x}_1^{lz}(i,1) = \begin{cases} 1, & \text{if } \ s_i \in \mathcal{S}^{lz} \\ \\ 0, & \text{if } \ s_i \notin \mathcal{S}^{lz} \end{cases} \in \mathcal{Z}_+^{n_s}, \quad i \in \{1,2,3,\ldots,n_s\} \tag{4.17}$$

In other words, $\mathbf{x}_1^{lz}$ represents the state vector of a resource type that lies in the sectors of $\mathcal{S}^{lz}$. The subscript "1" corresponds to the number of resource types similarly to the definitions of $\mathbf{x}_1^d$ and $\mathbf{x}_1^a$.

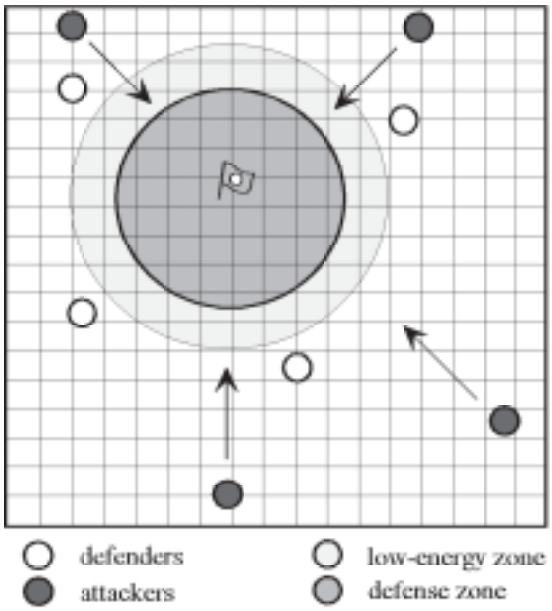


Figure 4.4: A "low-energy" zone is defined about the defense zone.

We also define $\mathbf{X}_1^{lz}$ as the state vector for the whole optimization horizon, N_p, i.e.,

$$\mathbf{X}_1^{lz} = \begin{pmatrix} \mathbf{x}_1^{lz}[1] \\ \mathbf{x}_1^{lz}[2] \\ \cdots \\ \mathbf{x}_1^{lz}[N_p] \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1^{lz} \\ \mathbf{x}_1^{lz} \\ \cdots \\ \mathbf{x}_1^{lz} \end{pmatrix} \in \mathcal{Z}_+^{N_p n_s} \tag{4.18}$$

Thus, the inner product $-\left(\mathbf{X}_1^{lz}\right)^T \cdot \mathbf{X}_1^d$ represents the defenders' cost for staying out of the low-energy zone. By defining the following linear objective function of the defenders' state vector

$$\text{minimize} \quad -w_a^d \cdot (\mathbf{X}_1^a)^T \cdot \mathbf{X}_1^d - w_{lz}^d \cdot \left(\mathbf{X}_1^{lz}\right)^T \cdot \mathbf{X}_1^d \tag{4.19}$$

or equivalently

$$\text{minimize} \quad -\left(w_a^d \cdot \mathbf{X}_1^a + w_{lz}^d \cdot \mathbf{X}_1^{lz}\right)^T \cdot \mathbf{X}_1^d \tag{4.20}$$

the defenders are able to decide about the weight they attach to getting closer to the low-energy zone, w_{lz}^d, or closer to the attackers, w_a^d. We can always take

$$w_a^d + w_{lz}^d = 1 \tag{4.21}$$

Hence, an alternative linear programming problem that can be used in deriving efficient paths for the defenders (according to the algorithm of Section 4.3.6) could be

$$\begin{aligned}
\text{minimize} \quad & -\left[\begin{array}{cc} \left(w_a^d \cdot \mathbf{X}_1^a + w_{lz}^d \cdot \mathbf{X}_1^{lz}\right)^T & \mathbf{0}^T \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{X}_1^d \\ \mathbf{U}_1^d \end{array} \right) \\
\text{subject to} \quad & \left[\begin{array}{cc} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_c^d - \mathbf{T}_{xu,c}^d \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{X}_1^d \\ \mathbf{U}_1^d \end{array} \right) \leq \left[\begin{array}{cc} \mathbf{T}_{xx,\mathrm{id}}^d & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_{xx_0,c}^d \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{1} \\ \mathbf{x}_1^d[0] \end{array} \right) \\
& \left[\begin{array}{cc} \mathbf{I} & -\mathbf{T}_{xu}^d \\ \left(\mathbf{X}_1^{dz}\right)^T & \mathbf{O} \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{X}_1^d \\ \mathbf{U}_1^d \end{array} \right) = \left[\begin{array}{cc} \mathbf{O} & \mathbf{T}_{xx_0}^d \\ \mathbf{O} & \mathbf{O} \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{1} \\ \mathbf{x}_1^d[0] \end{array} \right) \\
\text{variables} \quad & \mathbf{X}_1^d \in \Re_+^{N_p n_x^d}, \quad \mathbf{U}_1^d \in \Re_+^{N_p n_u^d}
\end{aligned} \tag{4.22}$$

4.3.9 Remarks

We showed that the problem of optimizing defense trajectories in the simplified version of the RoboFlag competition can be solved by a linear-programming-based path planning. This strategy was based on the linear-programming-based planning for allocating friendly resources in an adversarial environment. The only difference was the constraint that each resource level can take on only binary values. Although this problem is a *mixed integer programming problem*, it turned out that its *linear programming relaxation* in conjunction with a *receding horizon philosophy* was sufficient for providing us with a reliable path planning. In the following section we pursue a similar path planning for the attackers.

4.4 Linear-programming-based attack planning

The question that arises now is whether we are able to apply a path planning for the attackers similar to that applied for the defenders. The only difference between the attackers and the defenders is their objective. In particular, the attackers' objective is to infiltrate the defenders' protected zone, while in parallel they have to avoid being intercepted by the defenders.

Hence, we apply a similar procedure to that described in Section 4.3, according to which the playing area is divided into sectors, while each team (the defenders or the attackers) is viewed as a different resource type. Although the resource levels take on only binary values, this constraint can be relaxed as before. In this case, the constraints of both teams are linear, which means that we can derive a linear programming optimization problem for attack.

4.4.1 Linear programming relaxation

The attackers objective could be a weighted sum between the cost of staying out of the defense zone, $-\left(\mathbf{X}_1^{dz}\right)^T \cdot \mathbf{X}_1^a$, and the cost of getting closer to the defenders, $\left(\mathbf{X}_1^d\right)^T \cdot \mathbf{X}_1^a$. Let w_{dz}^a and w_d^a be the weights the attackers attach to getting closer to the defense zone and staying away from the defenders, respectively. We can always take

$$w_{dz}^a + w_d^a = 1 \tag{4.23}$$

Hence, an objective function for the attackers could be

$$\text{minimize} \quad -w_{dz}^a \cdot \left(\mathbf{X}_1^{dz}\right)^T \cdot \mathbf{X}_1^a + w_d^a \cdot \left(\mathbf{X}_1^d\right)^T \cdot \mathbf{X}_1^a \tag{4.24}$$

or equivalently

$$\text{minimize} \quad -\left(w_{dz}^a \cdot \mathbf{X}_1^{dz} - w_d^a \cdot \mathbf{X}_1^d\right)^T \cdot \mathbf{X}_1^a \tag{4.25}$$

Since the constraints for the attackers' resources are the same as those for the defenders' resources, except for the defense-zone constraints, the corresponding *mixed integer programming problem* for the attackers is similar to that of (4.10) and is given by the following equation:

$$\begin{aligned}
\text{minimize} \quad & -\left[\begin{array}{cc} \left(w_{dz}^a \cdot \mathbf{X}_1^{dz} - w_d^a \cdot \mathbf{X}_1^d\right)^T & \mathbf{0}^T \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{X}_1^a \\ \mathbf{U}_1^a \end{array} \right) \\
\text{subject to} \quad & \left[\begin{array}{cc} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_c^a - \mathbf{T}_{xu,c}^a \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{X}_1^a \\ \mathbf{U}_1^a \end{array} \right) \leq \left[\begin{array}{cc} \mathbf{T}_{xx,\text{id}}^a & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_{xx_0,c}^a \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{1} \\ \mathbf{x}_1^a[0] \end{array} \right) \\
& \left[\begin{array}{cc} \mathbf{I} & -\mathbf{T}_{xu}^a \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{X}_1^a \\ \mathbf{U}_1^a \end{array} \right) = \left[\begin{array}{cc} \mathbf{O} & \mathbf{T}_{xx_0}^a \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{1} \\ \mathbf{x}_1^a[0] \end{array} \right) \\
\text{variables} \quad & \mathbf{X}_1^a \in \mathcal{Z}_+^{N_p n_x^a}, \quad \mathbf{U}_1^a \in \Re_+^{N_p n_u^a}
\end{aligned} \tag{4.26}$$

If we relax the binary constraints $\mathbf{X}_1^a \in \mathcal{Z}_+^{N_p n_x^a}$ by setting $\mathbf{X}_1^a \in \Re_+^{N_p n_x^a}$, the resulting *linear programming relaxation* is

$$\begin{aligned}
\text{minimize} \quad & -\left[\begin{array}{cc} \left(w_{dz}^a \cdot \mathbf{X}_1^{dz} - w_d^a \cdot \mathbf{X}_1^d\right)^T & \mathbf{0}^T \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{X}_1^a \\ \mathbf{U}_1^a \end{array} \right) \\
\text{subject to} \quad & \left[\begin{array}{cc} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_c^a - \mathbf{T}_{xu,c}^a \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{X}_1^a \\ \mathbf{U}_1^a \end{array} \right) \leq \left[\begin{array}{cc} \mathbf{T}_{xx,\text{id}}^a & \mathbf{O} \\ \mathbf{O} & \mathbf{T}_{xx_0,c}^a \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{1} \\ \mathbf{x}_1^a[0] \end{array} \right) \\
& \left[\begin{array}{cc} \mathbf{I} & -\mathbf{T}_{xu}^a \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{X}_1^a \\ \mathbf{U}_1^a \end{array} \right) = \left[\begin{array}{cc} \mathbf{O} & \mathbf{T}_{xx_0}^a \end{array} \right] \cdot \left(\begin{array}{c} \mathbf{1} \\ \mathbf{x}_1^a[0] \end{array} \right) \\
\text{variables} \quad & \mathbf{X}_1^a \in \Re_+^{N_p n_x^a}, \quad \mathbf{U}_1^a \in \Re_+^{N_p n_u^a}
\end{aligned} \tag{4.27}$$

Recall that if an optimal solution to the relaxation is feasible for the mixed integer programming problem, it is also an optimal solution to its linear programming relaxation.

4.4.2 Computation of a suboptimal solution

In general, the solution of the *linear programming relaxation* is a non-integer vector, which means that this solution does not belong to the feasible set of the *mixed integer*

programming problem. However, as explained in Section 4.3.5, we are able to compute a suboptimal solution of the *mixed integer programming problem* of (4.26) by using an optimal solution of its *linear programming relaxation* of (4.27).

This suboptimal solution is computed by following a procedure similar to that applied in Section 4.3.5. In particular, given the optimal control of the *linear programming relaxation* for the whole optimization horizon, N_p,

$$(\mathbf{U}^*)^a = \left[\begin{array}{cccc} (\mathbf{u}^*)^a [t+0] & (\mathbf{u}^*)^a [t+1] & \cdots & (\mathbf{u}^*)^a [t+N_p] \end{array} \right]$$

we are only interested in the first optimal control, i.e., $(\mathbf{u}^*)^a [t] = (\mathbf{u}^*)^a [t+0]$, which will generally be a non-integer vector between $\mathbf{0}$ and $\mathbf{1}$.

Given this first optimal solution of the *linear programming relaxation*, we construct a *suboptimal solution* of the *mixed integer programming problem*. In particular, among the resource levels that exit from a sector, we pick up the maximum of them, to which we assign the value "1", while the rest of them are assigned the value "0". In this way, we define an integer control input that belongs to the feasible set of the *mixed integer programming problem* of (4.26), while, in parallel, the sum of the resource levels remains the same as that of the previous stage.

Algorithmically, according to the definition of the function $\mathcal{U}_{out} : \mathcal{SN}(\mathcal{S}, \mathcal{R}) \times \mathcal{R} \times \mathcal{S} \rightarrow \{1, 2, \ldots, n_u\}$ (see (2.12) in page 18), we define the corresponding function for the attackers' resources as

$$\mathcal{U}^a_{\text{out}} : \mathcal{SN}^a(\mathcal{S}, \mathcal{R}^a) \times \mathcal{R}^a \times \mathcal{S} \rightarrow \{1, 2, \ldots, n^a_u\} \tag{4.28}$$

where $\mathcal{SN}^a(\mathcal{S}, \mathcal{R}^a)$ is the set of neighboring sectors of the attackers' resources. This function maps a triple of the form (s_k, r_j, s_i), where $s_i \in \mathcal{S}$, $r_j \in \mathcal{R}^a$ and $s_k \in \mathcal{SN}^a(s_i, r_j)$, to the row number of the control vector $\mathbf{u}^a$ that corresponds to the control level $u^a_{s_k \leftarrow s_i, r_j}$, i.e.,

$$\mathbf{u}^a \left(\mathcal{U}^a_{\text{out}} \left(s_k, r_j, s_i \right), 1 \right) = u^a_{s_k \leftarrow s_i, r_j} \tag{4.29}$$

Let $\tilde{\mathbf{u}}^a[t]$ be a suboptimal solution of the mixed integer programming problem at time t. This solution can be constructed in the following way: For each sector $s_i \in \mathcal{S}$ and each resource type $r_j \in \mathcal{R}^a$,

1. Define the set

$$\mathcal{UN}^a(s_i, r_j) = \{\mathcal{U}^a_{\text{out}}(s_k, r_j, s_i) \;\; | \;\; s_k \in \mathcal{SN}^a(s_i, r_j)\}$$

that includes the row numbers of the control vector $\mathbf{u}^a$ which correspond to the control inputs $\left\{u^a_{s_k \leftarrow s_i, r_j} \;\; | \;\; s_k \in \mathcal{SN}^a(s_i, r_j)\right\}$.

2. If $x^a_{s_i,r_j} \neq 0$, find the row number, θ^*, of $(\mathbf{u}^*)^a[t]$ that corresponds to the maximum control level among those ones that exit from sector s_i, i.e.,

$$\theta^* = \arg \max_{\theta \in \mathcal{UN}^a(s_i, r_j)} \{(\mathbf{u}^*)^a[t](\theta, 1)\}$$

   (a) if $(\mathbf{u}^*)^a[t](\theta^*, 1) \geq 1 - \sum_{\theta \in \mathcal{UN}^a(s_i,r_j)} \{(\mathbf{u}^*)^a[t](\theta, 1)\}$, which means that the maximum control level that exits from sector s_i is greater than the resource level that remains at that sector, set

$$\tilde{\mathbf{u}}^a[t](\theta, 1) = \begin{cases} 1 & , \quad \text{if } \theta = \theta^* \\ 0 & , \quad \text{if } \theta \neq \theta^* \end{cases} , \quad \theta \in \mathcal{UN}^a(s_i, r_j)$$

   which implies that the resource level $x^a_{s_i,r_j}$ will move to the neighboring sector $s_k \in \mathcal{SN}^a(s_i, r_j)$ that satisfies $\mathcal{U}^a_{\text{out}}(s_k, r_j, s_i) = \theta^*$.

   (b) otherwise, set

$$\tilde{\mathbf{u}}^a[t](\theta, 1) = 0, \quad \forall \theta \in \mathcal{UN}^a(s_i, r_j).$$

   which implies that the resource level $x^a_{s_i,r_j}$ will remain at sector s_i.

3. If $x^a_{s_i,r_j} = 0$, set

$$\tilde{\mathbf{u}}^a[t](\theta, 1) = 0, \quad \forall \theta \in \mathcal{UN}^a(s_i, r_j).$$

The resulting integer control vector, $\tilde{\mathbf{u}}^a[t]$, belongs to the feasible set of the *mixed integer programming problem* of (4.26), while the sum of the resource levels remains the same as the one of the previous stage.

4.4.3 Control algorithm for attack

We computed a suboptimal solution of the *mixed integer programming problem*, (4.26), by solving only its *linear programming relaxation*, (4.27). This solution is implemented in a *receding horizon manner*. In particular, the linear-programming-based planning for the attackers is similar to the corresponding one for the defenders and contains the following steps:

1. Measure the new state vectors of the defenders, $\mathbf{x}^d$, and the attackers, $\mathbf{x}^a$.

2. Solve the *linear programming relaxation*, (4.27), of the *mixed integer programming problem* of (4.26). Let

$$(\mathbf{U}^*)^a = \left[\begin{array}{cccc} (\mathbf{u}^*)^a [t+0] & (\mathbf{u}^*)^a [t+1] & \cdots & (\mathbf{u}^*)^a [t+N_p] \end{array} \right]$$

   be its optimal control solution for the whole optimization horizon, N_p.

3. Given the first optimal control, $\mathbf{u}^a[t] = (\mathbf{u}^*)^a [t+0]$ of the *linear programming relaxation*, compute a suboptimal solution, $\tilde{\mathbf{u}}^a[t]$, for the *mixed integer programming problem*, according to Section 4.4.2.

4. Apply only the suboptimal solution, $\tilde{\mathbf{u}}^a[t]$.

5. Advance $t \leftarrow t+1$ and repeat.

Thus, by changing only the objective function of the linear programming path planning for the defenders, we derived a similar path planning for the attackers. Similarly, this planning can be transformed to include other objectives, such as avoidance of other stationary objects.

4.5 The enemy's feedback matrix

The path planning for defense and attack described in the previous sections, are based on a linear programming optimization problem in conjunction with a receding horizon philosophy. However, in both these optimization problems the objective function depends on the enemy's state vector for the whole optimization horizon, which is generally unknown to the friendly team. In this section, we explore the use of stochastic feedback matrices for the enemy team.

According to the model simplifications of Section 3.3.1, we assume that the enemy resources implements a feedback policy $\mathbf{G}^e$, i.e.,

$$(\mathbf{x}^e)^+ = \mathbf{x}^e + \mathbf{B}^e \cdot (\mathbf{G}^e \mathbf{x}^e) = (\mathbf{I} + \mathbf{B}^e \mathbf{G}^e) \cdot \mathbf{x}^e \tag{4.30}$$

As mentioned in Section 3.3.2, this feedback matrix can be used for creating probability maps for the enemy's future states. More specifically, we are able to create a feedback matrix $\mathbf{G}^e_{r_j}$ for each enemy resource type $r_j \in \mathcal{R}^e$. Finally, the total feedback matrix is defined as

$$\mathbf{G}^e = \begin{bmatrix} \mathbf{G}^e_{r_1} & \mathbf{O} & \dots & \mathbf{O} \\ \mathbf{O} & \mathbf{G}^e_{r_2} & \dots & \mathbf{O} \\ \dots & \dots & \dots & \dots \\ \mathbf{O} & \mathbf{O} & \dots & \mathbf{G}^e_{r_{n^e_r}} \end{bmatrix} \tag{4.31}$$

where n^e_r is the number of enemy resource types.

The enemy's state vector for the whole optimization horizon, $\mathbf{X}^e$, is given by

$$\mathbf{X}^e = \mathbf{T}^e_{xx_0,G} \cdot \mathbf{x}^e[0] \tag{4.32}$$

where $\mathbf{T}^e_{xx_0,G}$ was defined by equation (3.26).

In the following sections we create several stochastic feedback matrices that can be used either for defense or for attack. In each case, the creation of the stochastic feedback matrix is based on the possible objectives of the opposing team.

4.5.1 Stochastic feedback matrix for defense

The attackers' first priority is the infiltration of the defense zone, which means that their objective is to move closer to the defense zone. So, given the sector that an attacker lies in, it is "more likely" that the attacker (enemy) will move to a sector that is closer to the defense zone. For this reason, we define a probability measure

$$g^a : \{\mathcal{SN}^a(\mathcal{S}, \mathcal{R}^a) \cup \mathcal{S}\} \times \mathcal{R}^a \times \mathcal{S} \to [0, 1] \tag{4.33}$$

such that the probability that the attackers' resource type (unit) $r_j \in \mathcal{R}^a$ will move from sector $s_i \in \mathcal{S}$ to sector $s_k \in \mathcal{SN}^a(s_i, r_j)$ is

$$g^a_{s_k \leftarrow s_i, r_j} = \frac{\left[\beta\left(s_i, s_{dz}\right) + \sum_{s_n \in \mathcal{SN}^a(s_i, r_j)} \left\{\beta\left(s_n, s_{dz}\right)\right\}\right] - \beta\left(s_k, s_{dz}\right)}{\left[\bar{n}^a_{sn}(s_i, r_j) - 1\right] \cdot \left[\beta\left(s_i, s_{dz}\right) + \sum_{s_n \in \mathcal{SN}^a(s_i, r_j)} \left\{\beta\left(s_n, s_{dz}\right)\right\}\right]} \tag{4.34}$$

while the probability of staying at the same sector is

$$g^a_{s_i \leftarrow s_i, r_j} = \frac{\sum_{s_n \in \mathcal{SN}^a(s_i, r_j)} \left\{\beta\left(s_n, s_{dz}\right)\right\}}{\left[\bar{n}^a_{sn}(s_i, r_j) - 1\right] \cdot \left[\beta\left(s_i, s_{dz}\right) + \sum_{s_n \in \mathcal{SN}^a(s_i, r_j)} \left\{\beta\left(s_n, s_{dz}\right)\right\}\right]} \tag{4.35}$$

where s_{dz} is the sector that corresponds to the center of the defense zone, $\beta\left(s_k, s_{dz}\right)$ is the minimum distance (in sectors) from sector s_k to the center of the defense zone, and $\bar{n}^a_{sn}(s_i, r_j)$ is the number of neighboring sectors of sector $s_i \in \mathcal{S}$ including itself that can be reached by the attackers' resource type $r_j \in \mathcal{R}^a$ in one stage.

It is easily seen that

$$\begin{aligned} &(1) \quad g^a_{s_k \leftarrow s_i, r_j} \in [0, 1], \quad \forall s_i \in \mathcal{S}, \quad \forall r_j \in \mathcal{R}^a, \quad \forall s_k \in \{\mathcal{SN}^a(s_i, r_j) \cup s_i\} \\ &(2) \quad \sum_{s_k \in \mathcal{SN}^a(s_i, r_j)} g^a_{s_k \leftarrow s_i, r_j} + g^a_{s_i \leftarrow s_i, r_j} = 1, \quad \forall s_i \in \mathcal{S}, \quad \forall r_j \in \mathcal{R}^a \end{aligned} \tag{4.36}$$

which means that this probability measure satisfies the properties of (3.15). Therefore, according to (3.14) we can define a feedback matrix, $\mathbf{G}^a_{r_j}$, for each resource type, $r_j \in \mathcal{R}^a$.

The probability that an attacker will move to a neighboring sector, given by (4.34), or remain at the same sector, given by (4.35), depends only on the distance from that

sector to the defense zone. Since the position of the defense zone does not change over time, these probabilities do not also change over time. In other words, for each sector in the arena *the probabilities of (4.34) and (4.35) are constant over time and can be computed off-line.*

Furthermore, we can define a feedback matrix, $\mathbf{G}^a_{r_j}$, for each resource type of the attackers $r_j \in \mathcal{R}^a$, so that it includes the transition probabilities $g^a_{s_k \leftarrow s_i, r_j}$, given by (4.34), for all $s_i \in \mathcal{S}$ and $s_k \in \mathcal{SN}^a(s_i, r_j)$, i.e.,

$$\mathbf{G}^a_{r_j}\left(\mathcal{U}^a_{\text{out}}(s_k, r_j, s_i), s_i\right) = g^a_{s_k \leftarrow s_i, r_j}, \quad \forall s_i \in \mathcal{S}, \quad \forall s_k \in \mathcal{SN}^a(s_i, r_j) \tag{4.37}$$

where $\mathcal{U}^a_{\text{out}} : \mathcal{SN}^a(\mathcal{S}, \mathcal{R}^a) \times \mathcal{R}^a \times \mathcal{S} \rightarrow \{1, 2, \ldots, n^a_u\}$ is the function that maps a triple of the form (s_k, r_j, s_i), where $s_i \in \mathcal{S}$, $r_j \in \mathcal{R}^a$ and $s_k \in \mathcal{SN}^a(s_i, r_j)$, to the row number of $\mathbf{u}^a$ that corresponds to $u^a_{s_k \leftarrow s_i, r_j}$. The probability that the defender stays at the same sector, $g^a_{s_i \leftarrow s_i, r_j}$, is not included directly in the feedback matrix $\mathbf{G}^a_{r_j}$ since from (4.36) we have that

$$g^a_{s_i \leftarrow s_i, r_j} = 1 - \sum_{s_k \in \mathcal{SN}^a(s_i, r_j)} g^a_{s_k \leftarrow s_i, r_j} \tag{4.38}$$

According to this definition of stochastic feedback matrix $\mathbf{G}^a_{r_j}$, given any actual state vector of the attackers we can compute the probability distribution of their resource levels for more than one stages ahead. This property is very useful, since we prefer to optimize the defenders' paths for $N_p > 1$ stages ahead.

Summarizing, the advantages of this feedback matrix are the following:

- It includes all transition probabilities for each sector in the arena.
- It can be computed off-line.
- It can be used in computing the probability distribution of the attackers' resource levels for more than one stage ahead.

Figure 4.6 (page 80) shows the probability distribution of the attackers' resource levels after $N_p = 2$ stages when the actual state vector includes three attackers. We should expect that the probability that an attacker moves closer to the defense zone is higher than the probability of moving away from the defense zone. Although this is the case, according to the definition of $\mathbf{G}^a_{r_j}$, the difference in the size between these probabilities is quite small and cannot be easily distinguished in Figure 4.6.

4.5.2 Alternative stochastic feedback matrix for defense

Alternatively, let us define $\bar{n}^a_{dsn}(s_i, r_j)$ as the number of neighboring sectors of sector s_i including itself that can be reached by the attackers' resource type (unit) r_j and are the most probable next positions for that type based on their distance from the defense zone. In other words, instead of considering all the neighboring sectors of s_i as possible next positions of the attackers' resource type r_j, we take into account a smaller number of neighboring sectors that are the closest ones to the defense zone. This number of neighboring sectors of s_i is of our choice, and it can be chosen to be the same for every sector $s_i \in \mathcal{S}$ and resource type $r_j \in \mathcal{R}^a$, i.e.,

$$\bar{n}^a_{dsn}(s_i, r_j) = \bar{n}^a_{dsn}, \quad \forall s_i \in \mathcal{S}, \quad \forall r_j \in \mathcal{R}^a \tag{4.39}$$

Let also $\mathcal{DSN}^a(s_i, r_j, \bar{n}^a_{dsn})$ be the set of these sectors. In this case, we can define the following probability measure:

$$g^a_{\text{alt}} : \{\mathcal{DSN}^a(\mathcal{S}, \mathcal{R}^a, \bar{n}^a_{dsn})\} \times \mathcal{R}^a \times \mathcal{S} \to [0, 1] \tag{4.40}$$

such that the probability that the attackers' resource type (unit) $r_j \in \mathcal{R}^a$ will move from sector $s_i \in \mathcal{S}$ to $s_k \in \mathcal{DSN}^a(s_i, r_j, \bar{n}^a_{dsn})$ is

$$g^a_{s_k \leftarrow s_i, r_j, \text{alt}} = \frac{\left[\beta\left(s_i, s_{dz}\right) + \sum_{s_n \in \mathcal{DSN}^a(s_i, r_j, \bar{n}^a_{dsn})} \left\{\beta\left(s_n, s_{dz}\right)\right\}\right] - \beta\left(s_k, s_{dz}\right)}{\left(\bar{n}^a_{dsn} - 1\right) \cdot \left[\beta\left(s_i, s_{dz}\right) + \sum_{s_n \in \mathcal{DSN}^a(s_i, r_j, \bar{n}^a_{dsn})} \left\{\beta\left(s_n, s_{dz}\right)\right\}\right]} \tag{4.41}$$

Similarly to Section 4.5.1 we can define a feedback matrix, $\mathbf{G}^a_{r_j,\text{alt}}$, for each resource type $r_j \in \mathcal{R}^a$ of the attackers, so that it includes the transition probabilities,

$g^a_{s_k \leftarrow s_i, r_j, \text{alt}}$, given by equation (4.41) for all $s_i \in \mathcal{S}$ and $s_k \in \mathcal{DSN}^a(s_i, r_j, \bar{n}^a_{dsn})$, i.e.,

$$\begin{gathered} \mathbf{G}^a_{r_j,\text{alt}}\left(\mathcal{U}^a_{\text{out}}(s_k, r_j, s_i), s_i\right) = g^a_{s_k \leftarrow s_i, r_j, \text{alt}}, \\ \forall s_i \in \mathcal{S}, \quad \forall s_k \in \mathcal{DSN}\left(s_i, r_j, \bar{n}^a_{dsn}\right) \setminus \{s_i\} \end{gathered} \tag{4.42}$$

The probability that the defender stays at the same sector, $g^a_{s_i \leftarrow s_i, r_j, \text{alt}}$, is not included directly in the feedback matrix $\mathbf{G}^a_{r_j,\text{alt}}$ because

$$g^a_{s_i \leftarrow s_i, r_j, \text{alt}} = 1 - \sum_{s_k \in \mathcal{DSN}^a\left(s_i, r_j, \bar{n}^a_{dsn}\right) \setminus \{s_i\}} g^a_{s_k \leftarrow s_i, r_j, \text{alt}} \tag{4.43}$$

This feedback matrix has the same advantages with $\mathbf{G}^a_{r_j}$ computed in the previous section. In other words, $\mathbf{G}^a_{r_j,\text{alt}}$ can include the transition probabilities of the attackers for the whole optimization horizon N_p, while these probabilities can be computed off-line. Figure 4.7 (page 81) shows the probability distribution for several attackers and for $N_p = 2$, $\bar{n}^a_{dsn} = 6$.

4.5.3 Stochastic feedback matrix for attack

The defenders' first priority is the interception of the attackers, which means that their objective is to move closer to the attackers. So, given the sector that a defender lies in, it is more likely that a defender (enemy) will move to a sector that is closer to the next most probable positions (for the whole optimization horizon) of an attacker (friend).

In particular, we could define a probability measure similar to that of Section 4.5.1 that is conditional on the distances from each defender to the most probable next positions of each attacker. For this reason, this probability measure depends on the current state vector of both defenders and attackers, which means that it cannot be computed off-line. In other words, at each stage and given the current positions of the attackers and the defenders we have to update the probability distribution of the defenders.

Let us define $\mathbf{s}^a_{mp,t} \in \mathcal{N}^{n^a_r}$ as the vector of the most probable positions (sectors) of the attackers after t time intervals, i.e., each entry of this vector is given by

$$\mathbf{s}^a_{mp,t}(q,1) = \arg\max_{s_i \in \mathcal{S}} \left\{ g^a_{s_i,r_q,t} \right\}, \quad q \in \{1,2,\ldots,n^a_r\} \tag{4.44}$$

where $g^a_{s_i,r_q,t}$ is the probability that the attackers' resource type $r_q \in \mathcal{R}^a$ is at sector s_i after t time intervals.

Note that this position is computed by the attackers (friendly team) and is an estimation of what the defenders (enemy team) "believe" about the most probable next positions of the attackers. One way to compute $g^a_{s_i,r_q,t}$ is to use a version of the stochastic matrix "$\mathbf{G}^a$", computed in Sections 4.5.1 and 4.5.2, that the defenders might use.

Given these most probable next positions of the attackers, for each future stage we define a probability measure for the defenders as

$$g^d_t : \left\{ \mathcal{SN}^d(\mathcal{S}, \mathcal{R}^d) \cup \mathcal{S} \right\} \times \mathcal{R}^d \times \mathcal{S} \to [0,1] \tag{4.45}$$

such that the probability that after t time intervals the defenders' resource type (unit) $r_j \in \mathcal{R}^d$ moves from sector $s_i \in \mathcal{S}$ to sector $s_k \in \mathcal{SN}^d(s_i, r_j)$, provided that $g^d_{s_i,r_j,t} \neq 0$, is

$$g^d_{s_k \leftarrow s_i,r_j,t} = \frac{\left[\beta\left(s_i, \mathbf{s}^a_{mp,t}\right) + \sum_{s_n \in \mathcal{SN}^d(s_i,r_j)} \left\{\beta\left(s_n, \mathbf{s}^a_{mp,t}\right)\right\}\right] - \beta\left(s_k, \mathbf{s}^a_{mp,t}\right)}{[\bar{n}^d_{sn}(s_i,r_j) - 1] \cdot \left[\beta\left(s_i, \mathbf{s}^a_{mp,t}\right) + \sum_{s_n \in \mathcal{SN}^d(s_i,r_j)} \left\{\beta\left(s_n, \mathbf{s}^a_{mp,t}\right)\right\}\right]} \tag{4.46}$$

while the probability of remaining at sector s_i is

$$g^d_{s_k \leftarrow s_i,r_j,t} = \frac{\sum_{s_n \in \mathcal{SN}^d(s_i,r_j)} \left\{\beta\left(s_n, \mathbf{s}^a_{mp,t}\right)\right\}}{[\bar{n}^d_{sn}(s_i,r_j) - 1] \cdot \left[\beta\left(s_i, \mathbf{s}^a_{mp,t}\right) + \sum_{s_n \in \mathcal{SN}^d(s_i,r_j)} \left\{\beta\left(s_n, \mathbf{s}^a_{mp,t}\right)\right\}\right]} \tag{4.47}$$

where

$$\beta\left(s_i, \mathbf{s}^a_{mp,t}\right) = \min_{q \in \{1,2,\ldots,n^a_r\}} \beta\left(s_i, \mathbf{s}^a_{mp,t}(q,1)\right), \quad s_i \in \mathcal{S} \tag{4.48}$$

is the minimum distance (in sectors) from sector s_i to the most probable positions of the attackers after t time intervals.

In case $g^d_{s_i,r_j,t} = 0$, i.e., the probability that after t time intervals there is resource type $r_j \in \mathcal{R}^d$ at sector $s_i \in \mathcal{S}$ is zero, we define

$$g^d_{s_k \leftarrow s_i,r_j,t} = 0, \quad \forall s_k \in \{\mathcal{SN}^d(s_i, r_j) \cup s_i\} \tag{4.49}$$

Furthermore, we can define a feedback matrix, $\mathbf{G}^d_{r_j}$, for each resource type of the defenders $r_j \in \mathcal{R}^d$, so that it includes the transition probabilities $g^d_{s_k \leftarrow s_i,r_j,t}$ for all $s_i \in \mathcal{S}$ and $s_k \in \mathcal{SN}^d(s_i, r_j)$ and for all future stages of the optimization horizon $t \in \{1, 2, \ldots, N_p\}$, i.e.,

$$\begin{gathered}\mathbf{G}^d_{r_j}\left(\mathcal{U}^d_{\text{out}}(s_k, r_j, s_i), s_i\right) = g^d_{s_k \leftarrow s_i,r_j,t}, \quad \forall s_i \in \mathcal{S}, \quad \forall s_k \in \mathcal{SN}^d(s_i, r_j), \\ \forall t \in \{1, 2, \ldots, N_p\}\end{gathered} \tag{4.50}$$

where $\mathcal{U}^d_{\text{out}} : \mathcal{SN}^d(\mathcal{S}, \mathcal{R}^d) \times \mathcal{R}^d \times \mathcal{S} \rightarrow \{1, 2, \ldots, n^d_u\}$ is the function that maps a triple of the form (s_k, r_j, s_i), where $s_i \in \mathcal{S}$, $r_j \in \mathcal{R}^d$ and $s_k \in \mathcal{SN}^d(s_i, r_j)$ to the row number of $\mathbf{u}^d$ that corresponds to $u^d_{s_k \leftarrow s_i,r_j}$. The probability that the defender stays at the same sector, $g^d_{s_i \leftarrow s_i,r_j,t}$, is not included directly in the feedback matrix $\mathbf{G}^d_{r_j}$ since for each future stage $t \in \{1, 2, \ldots, N_p\}$ the following equation holds:

$$g^d_{s_i \leftarrow s_i,r_j,t} = 1 - \sum_{s_k \in \mathcal{SN}^d(s_i,r_j)} g^d_{s_k \leftarrow s_i,r_j,t} \tag{4.51}$$

Figure 4.8 (page 82) shows the probability distribution of the positions of two defenders after two stages when two attackers are close to them.

4.5.4 Velocity-based stochastic feedback matrix

An alternative stochastic feedback matrix, $\mathbf{G}^v$, can be defined by computing the previous and current direction of motion of the enemy resources. In particular, the Figure 4.5 shows the possible directions of motion in an arena of sectors.

If we assume that the distance between any two neighboring sectors is equal to 1, then each of the above directions of motion can be described by a vector. In

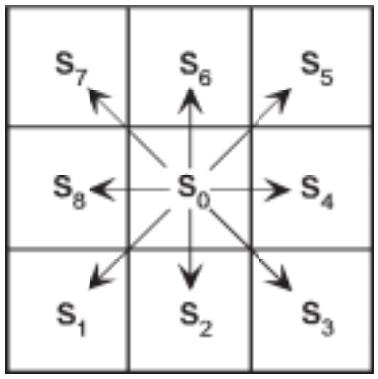


Figure 4.5: Possible directions of motion in an arena of sectors.

particular, according to Figure 4.5, we can define the following directions of motion:

$$\begin{gathered}\mathbf{v}_{s_0} = (0,0), \quad \mathbf{v}_{s_1} = (-1,-1), \quad \mathbf{v}_{s_2} = (0,-1), \quad \mathbf{v}_{s_3} = (1,-1), \\ \mathbf{v}_{s_4} = (1,0), \quad \mathbf{v}_{s_5} = (1,1), \quad \mathbf{v}_{s_6} = (0,1), \quad \mathbf{v}_{s_7} = (-1,1), \quad \mathbf{v}_{s_8} = (-1,0)\end{gathered} \tag{4.52}$$

where $\mathbf{v}_{s_i}$ denotes the direction of motion from the current position, s_0, to the next position, s_i.

Let $\mathbf{v}_{s_i,r_j}$ be the current direction of motion of resource type $r_j \in \mathcal{R}^e$ that lies in sector $s_i \in \mathcal{S}$ and $\mathbf{v}^+_{s_k,r_j}$ be the next direction of motion of the same resource type that moves from s_i to $s_k \in \{\mathcal{SN}^e(s_i, r_j) \cup s_i\}$. Then we can define a probability measure that assigns a probability to the inner product of the current and next direction of motion, i.e.,

$$g^v : \left\{ \langle \mathbf{v}_{s_i,r_j}, \mathbf{v}^+_{s_k,r_j} \rangle \;\middle|\; s_i \in \mathcal{S}, \quad r_j \in \mathcal{R}^e, \quad s_k \in \{\mathcal{SN}^e(s_i, r_j) \cup s_i\} \right\} \rightarrow [0,1] \tag{4.53}$$

Generally speaking, it is more likely that each enemy will retain the same direction of motion. In other words, the greater the inner product, the greater the probability that corresponds to this direction of motion.

This probability measure depends on the dynamics of the vehicles. For example, if the mass of a vehicle is great enough, then the probability that the direction of its motion will be reversed at the next stage is small. Since this analysis is not restricted

in any specific multi-vehicle system, we chose an arbitrary g^v, which is given by

$$g^v_{s_k \leftarrow s_i, r_j} = \begin{cases} 0 & \text{, if } \langle \mathbf{v}_{s_i,r_j}, \mathbf{v}^+_{s_k,r_j} \rangle < 0 \\ 1/12 & \text{, if } \langle \mathbf{v}_{s_i,r_j}, \mathbf{v}^+_{s_k,r_j} \rangle = 0 \\ 1/4 & \text{, if } \langle \mathbf{v}_{s_i,r_j}, \mathbf{v}^+_{s_k,r_j} \rangle > 0 \end{cases}, \tag{4.54}$$

$$s_i \in \mathcal{S}, \quad r_j \in \mathcal{R}^e, \quad s_k \in \{\mathcal{SN}^e(s_i, r_j) \cup s_i\}$$

This probability measure assigns greater probability to a positive inner product, which corresponds to a small change in the direction of motion, than to a negative one, which corresponds to a great change in the direction of motion.

It is easily seen that this probability measure satisfies the following properties:

$$\begin{aligned} &(1) \quad g^v_{s_k \leftarrow s_i, r_j} \in [0,1], \quad \forall s_i \in \mathcal{S}, \quad \forall r_j \in \mathcal{R}^e, \quad \forall s_k \in \{\mathcal{SN}^e(s_i, r_j) \cup s_i\} \\ &(2) \quad \textstyle\sum_{s_k \in \{\mathcal{SN}^e(s_i,r_j) \cup s_i\}} g^v_{s_k \leftarrow s_i, r_j} = 1, \quad \forall s_i \in \mathcal{S}, \quad \forall r_j \in \mathcal{R}^e \end{aligned} \tag{4.55}$$

which means that this probability measure satisfies the properties of (3.15). Therefore, according to (3.14), we can define a feedback matrix, $\mathbf{G}^v_{r_j}$, for each enemy resource type $r_j \in \mathcal{R}^e$. This feedback matrix includes the transition probabilities $g^v_{s_k \leftarrow s_i, r_j}$, given by (4.54), for all $s_i \in \mathcal{S}$ and $s_k \in \mathcal{SN}^e(s_i, r_j)$, i.e.,

$$\mathbf{G}^v_{r_j}\left(\mathcal{U}^e_{\text{out}}(s_k, r_j, s_i), s_i\right) = g^v_{s_k \leftarrow s_i, r_j}, \quad \forall s_i \in \mathcal{S}, \quad \forall s_k \in \mathcal{SN}^e(s_i, r_j) \tag{4.56}$$

where $\mathcal{U}^e_{\text{out}} : \mathcal{SN}^e(\mathcal{S}, \mathcal{R}^e) \times \mathcal{R}^e \times \mathcal{S} \to \{1, 2, \ldots, n^e_u\}$ is the function that maps a triple of the form (s_k, r_j, s_i), where $s_i \in \mathcal{S}$, $r_j \in \mathcal{R}^e$ and $s_k \in \mathcal{SN}^e(s_i, r_j)$, to the row number of $\mathbf{u}^e$ that corresponds to $u^e_{s_k \leftarrow s_i, r_j}$. The probability that the defender stays at the same sector, $g^v_{s_i \leftarrow s_i, r_j, t}$, is not included directly in the feedback matrix $\mathbf{G}^v_{r_j}$ since from (4.55) we have that

$$g^v_{s_i \leftarrow s_i, r_j} = 1 - \sum_{s_k \in \mathcal{SN}^e(s_i, r_j)} g^v_{s_k \leftarrow s_i, r_j} \tag{4.57}$$

According to this definition of the stochastic feedback matrix $\mathbf{G}^v_{r_j}$, given any actual state vector of the enemy we can compute the probability distribution of the enemy

resource levels for only one stage ahead, which is one of the disadvantages of this feedback matrix. Moreover, since the probabilities are conditional on the previous direction of motion, this feedback matrix cannot be computed off-line.

4.5.5 Combined stochastic feedback matrix

Both defenders and attackers feedback matrices, $\mathbf{G}^a$ and $\mathbf{G}^d$, can be combined with a velocity-based feedback matrix, $\mathbf{G}^v$. In particular, we could define a new probability matrix for the attackers, let $\mathbf{G}^c$, as

$$\mathbf{G}^c = \left(\mathbf{G}^i + \mathbf{G}^v\right)/2, \quad i \in \{a, d\} \tag{4.58}$$

4.6 Remarks

In this chapter we explored the utility of the linear-programming-based planning for resource allocation, described in the Chapter 3, in deriving optimal paths for multi-vehicle control systems with adversaries. In particular, we considered the RoboFlag competition which involves two teams of robots with opposing interests, the defenders and the attackers. We derived control algorithms for both defense and attack that are based on the linear-programming-based planning for resource allocation. In this case, both defenders and attackers were modelled as different resource types in an arena of sectors. Moreover, since adversaries are modelled as state-dependent, various stochastic feedback laws based on their current position and/or velocity can describe their possible future attitude, which was included in the optimization.

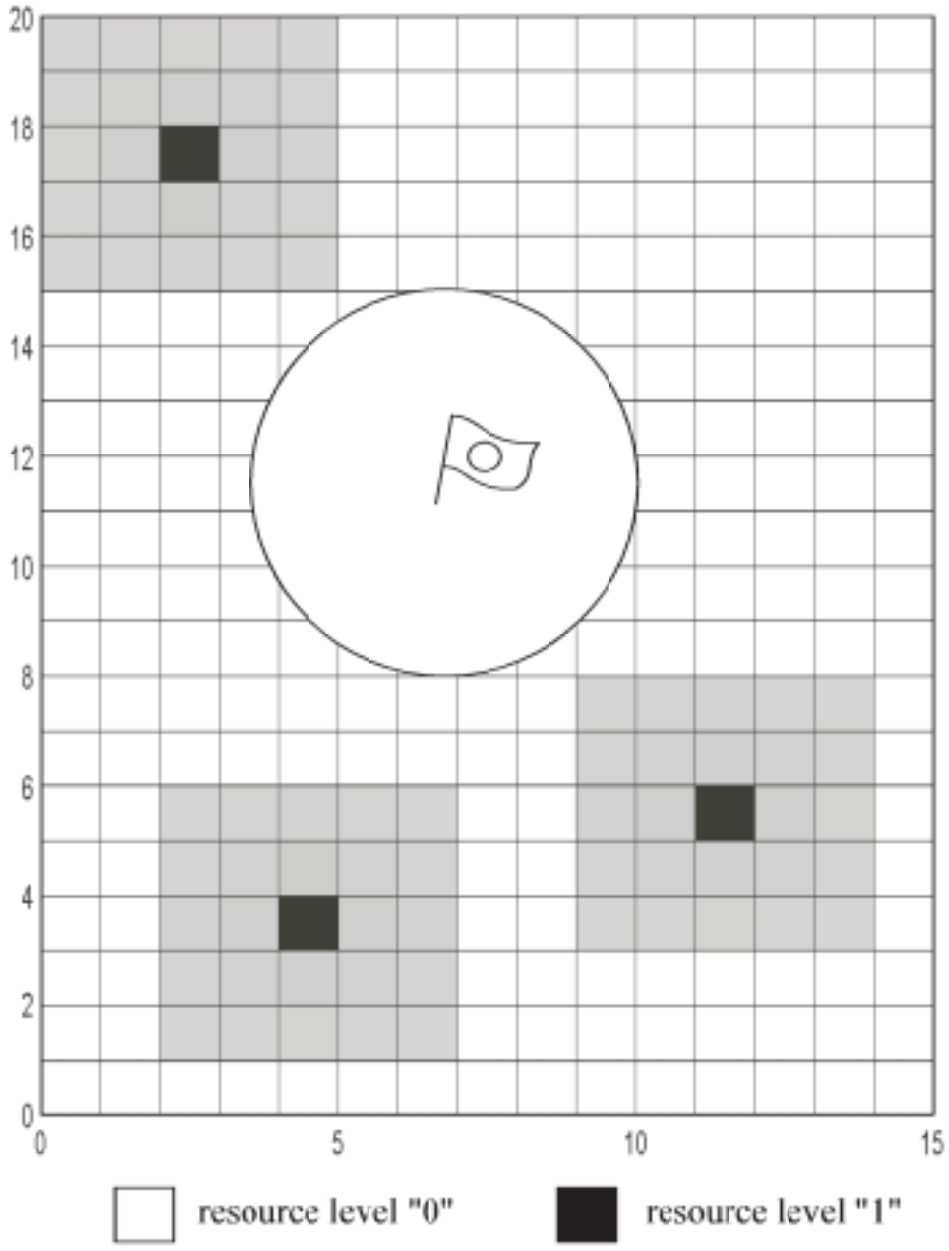


Figure 4.6: Probability distribution of three attackers after $N_p = 2$ stages based on the stochastic feedback matrix for defense.

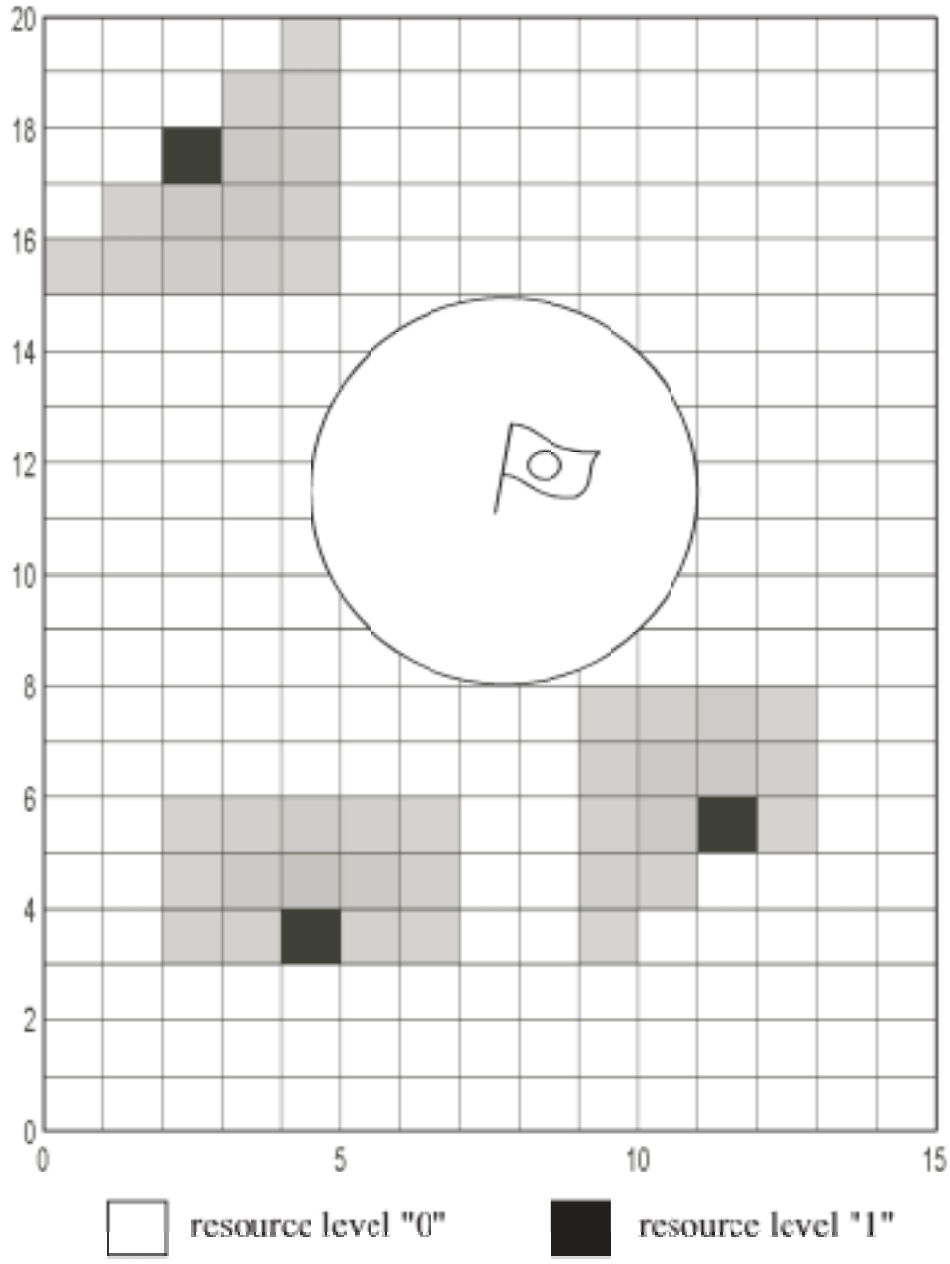


Figure 4.7: Probability distribution of three attackers after $N_p = 2$ stages based on the alternative stochastic feedback matrix for defense with $\bar{n}^a_{dsn} = 6$.

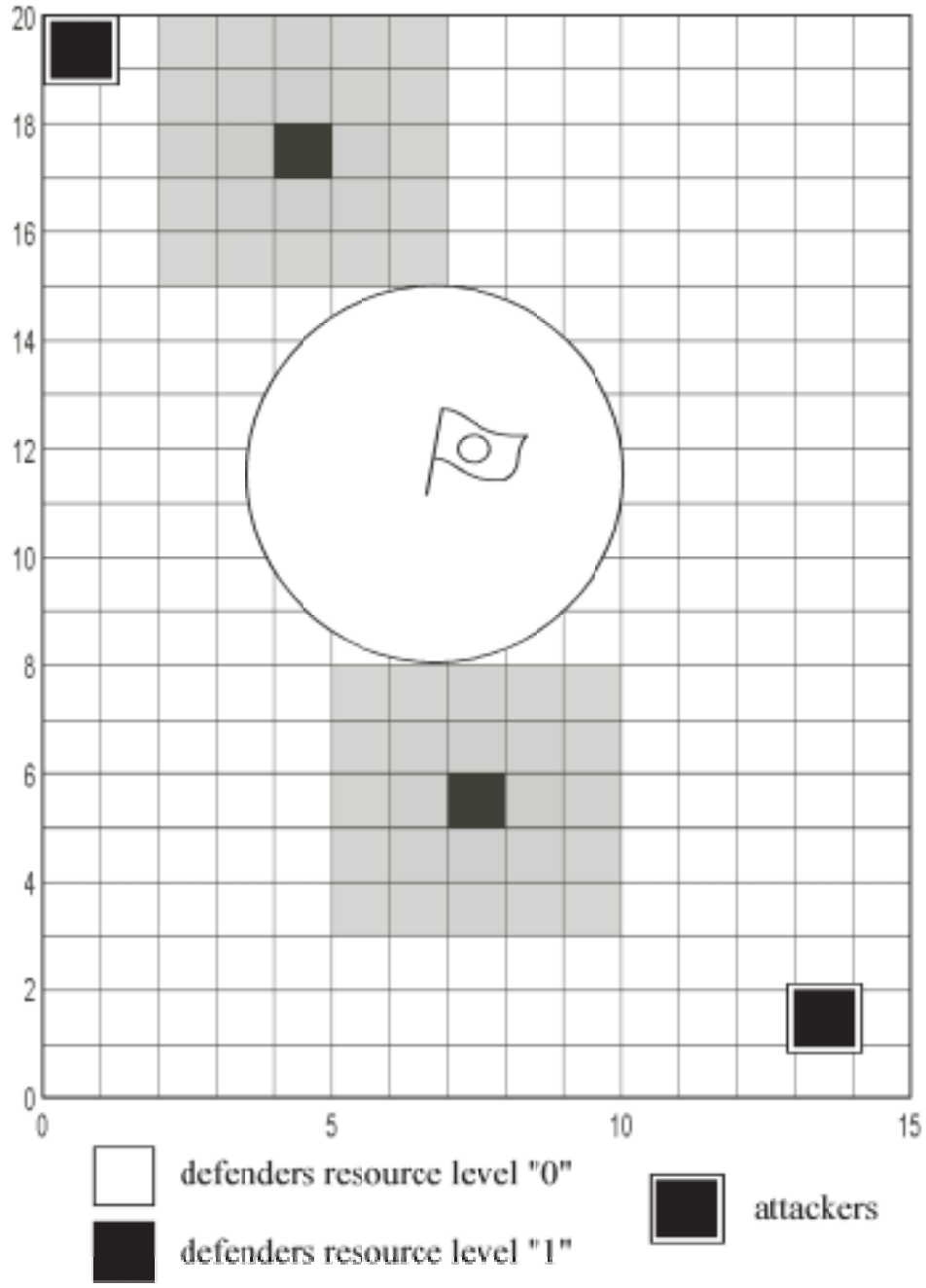


Figure 4.8: Probability distribution of two defenders after $N_p = 2$ stages based on the stochastic feedback matrix for attack.

CHAPTER 5

Simulations

5.1 Introduction

In this chapter, the multi-vehicle path planning for defense and attack derived for the RoboFlag competition in Chapter 4 are applied to the "Cornell RoboFlag Simulator" and to a RoboFlag simulator created in Matlab. Our objective is to test the linear-programming-based path planning for both defense and attack derived in Sections 4.3 and 4.4, respectively. To this end, we design several versions of the original RoboFlag competition and we test these plans for both their effectiveness and computational efficiency.

5.2 The "Cornell RoboFlag Simulator"

The architecture of the "Cornell RoboFlag Simulator" is described in detail by D'Andrea and Babish in [11], while the complete documentation is given in [33]. This simulator is created in C++ and designed to model the RoboFlag competition between two teams of robots described by D'Andrea and Murray in [12]. Here we consider a simpler version of the RoboFlag competition where one of the teams always attack the other team's protected zone (defense zone), while the latter one always defends. This version can also be modelled by the "Cornell RoboFlag Simulator" and consists of four main parts:

- The arbiter

- The defenders' computer
- The attackers' computer
- The robots

The arbiter is responsible for deciding about the current state of each robot (active or inactive). In particular, a robot becomes inactive when it is intercepted by an opponent. Moreover, it sends the current data of the game (positions, velocities and state of the robots) to both defenders and attackers.

The computer of each team is responsible for computing the next destinations of robots (high control level). These new destinations are sent to the robots through the communications network. When a robot receives its new destination it tracks the line segment between its current position with its next destination (low control level). A general architecture of the "Cornell RoboFlag Simulator" is given in Figure 5.1 (page 85).

The messaging system of this simulator is similar to that of a real multi-vehicle system, because it includes a lag between the time a message is sent and the time a message is received, while there is a small probability that a message can be dropped. In addition the size of the messages is limited by a simulated bandwidth.

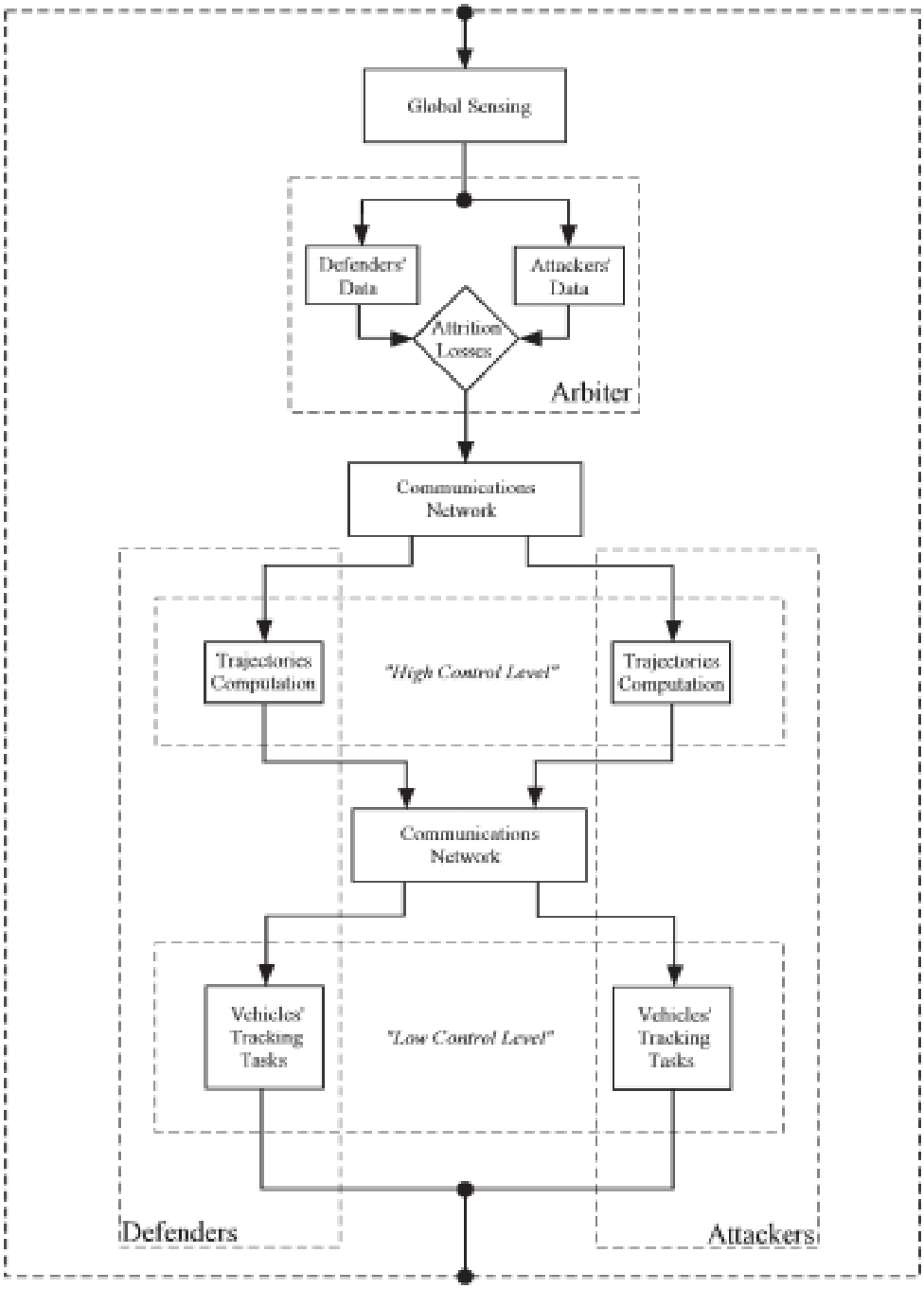


Figure 5.1: A general architecture of the "Cornell RoboFlag Simulator."

5.3 A RoboFlag simulator created in Matlab

We also created a RoboFlag simulator in Matlab. This simulator models the simplified version of the RoboFlag competition described in the previous section. However, the messaging system of the "Cornell RoboFlag Simulator" is not included, i.e., there is no lag between the time a message is sent and the time a message is received, while the probability that a message can be dropped is zero. Finally, the size of the messages is not limited by any simulated bandwidth.

Due to these differences, this simulator is simpler than the "Cornell RoboFlag Simulator," because every message is sent it will be received with probability one and without any time delay. For this reason, this simulator will be used in comparing the several path planning algorithms derived in the previous chapter. On the other hand, since the "Cornell RoboFlag Simulator" is more realistic than the one created in Matlab, we will also test the linear-programming-based path planning in this simulator.

5.4 Simulations and Discussion

Several path planning algorithms based on linear programming were tested in both simulators described in the previous sections. In particular we tested both the control algorithm for defense described in Section 4.3.6 and the control algorithm for attack described in Section 4.4.3.

In Figure 5.2 the defenders implement a control algorithm for defense according to Section 4.3.6 in the RoboFlag simulator created in Matlab. In parallel, a stochastic feedback matrix for defense is used according to Section 4.5.2 with $\bar{n}^a_{dsn} = 6$. On the other hand, the attackers follow pre-specified paths that are unknown to the defenders. It is easily seen, that although defenders do not know the attackers' paths,

they can predict accurately the next positions of the attackers and intercept them.

This game was also simulated in the "Cornell RoboFlag Simulator" which is more realistic than the one created in Matlab. The resulting paths of the defenders are shown in Figure 5.3. Even though there is a small lag between the time a message is sent and the time it is received, the defenders are able to intercept the attackers. Moreover, this control algorithm for defense is also computationally efficient since it runs in 3 seconds for three attackers, an arena of 300 sectors and an optimization horizon of $N_p = 6$ stages. This algorithm can run even faster if we reduce the number of sectors or the optimization horizon.

In order to check the utility of a velocity-based feedback matrix, we extend the previously described game. More specifically, now the defenders use a combined stochastic feedback matrix for defense that includes both a feedback matrix according to Section 4.5.2 with $\bar{n}^a_{dsn} = 6$ and a velocity-based feedback matrix according to Section 4.5.4. Again the attackers use pre-specified paths that are unknown to the defenders. The resulting paths of the defenders are shown in Figure 5.4. In comparison with Figure 5.2, we noticed that in this case some defenders move faster towards the attackers due to the fact that their prediction about the attackers' next positions is more accurate.

Since the linear-programming-based path planning for defense was satisfactory we are interested in testing this algorithm against smarter attackers. To this end, the attackers implement a control algorithm for attack based on Section 4.4.3 and with optimization horizon $N_p = 6$. The weight that the attackers attach to getting closer to the defense zone is chosen to be $w^a_{dz} = 0.01$, which means that $w^a_d = 0.99$. The attackers also use a stochastic feedback matrix for attack based on Section 4.5.3 in order to predict the defenders' future positions. On the other hand, the defenders implement a control algorithm for defense according to Section 4.3.6 with $N_p = 6$, in combination with a stochastic feedback matrix for defense based on Section 4.5.2

with $\bar{n}^a_{dsn} = 6$.

This game was tested in the RoboFlag simulator created in Matlab and the resulting paths are shown in Figure 5.5. It is easily seen that the attackers try both to avoid the defenders and at the same time to get closer to the defense zone. However, only one attacker was finally succeed to infiltrate the defense zone which is quite reasonable since the defenders implement an optimization path planning for defense.

The question that arises now is whether attackers can do better than that. For this reason, we implement a slightly different control algorithm for attack while defenders use the same path planning as in the previous case. In particular, the attackers use weights $w^a_{dz} = 0.01$ and $w^a_d = 0.99$ only at the first and last stage of the optimization horizon, while at any other stage $w^a_{dz} = 0$ and $w^a_d = 1$, which means that they attach more weight on avoiding the defenders than getting closer to the defense zone. The resulting paths are shown in Figure 5.6.

Although only one attacker infiltrated the defense zone, it is obvious that the attackers attach larger weight to avoiding the defenders in comparison with the previous simulation. On the other hand, the defenders are not able in both Figure 5.5 and Figure 5.6 to intercept more than two attackers. Thus, it is reasonable to ask whether the defenders can intercept more attackers or keep them away from the defense zone.

Trying to answer this question an alternative objective function is implemented for the defenders which was described in Section 4.3.8. In particular, not only do the defenders try to get closer to the attackers, but also try to stay closer to the defense zone. We assume that the weight they attach to getting closer to the attackers is $w^d_a = 0.95$, which means that the weight they attach to getting closer to the defense zone is $w^d_{lz} = 0.05$. The resulting paths are shown in Figure 5.7.

Although one attacker infiltrated the defense zone, it is easily seen that defenders optimization path planning is better than in the previous cases. In particular, the defenders make the attackers follow longer paths towards the defense zone, which

means that it is more difficult for the attackers to find a clear path to the defense zone.

5.5 Remarks

The preceding simulations showed that a linear-programming-based path planning can be used for either defense or attack in the RoboFlag competition. In particular, the control algorithm for defense (Section 4.3.6) in combination with a stochastic feedback matrix for defense (Section 4.5.2) and/or a velocity-based stochastic feedback matrix (Section 4.5.4) has satisfactory results against "smart" attackers. Moreover, an alternative objective function for defense which also includes the cost of staying away from the defense zone enforces attackers to follow longer paths towards the defense zone. On the other hand, the control algorithm for attack (Section 4.4.3) guarantees that a number of attackers infiltrate the defense zone independently of the defenders' path planning.

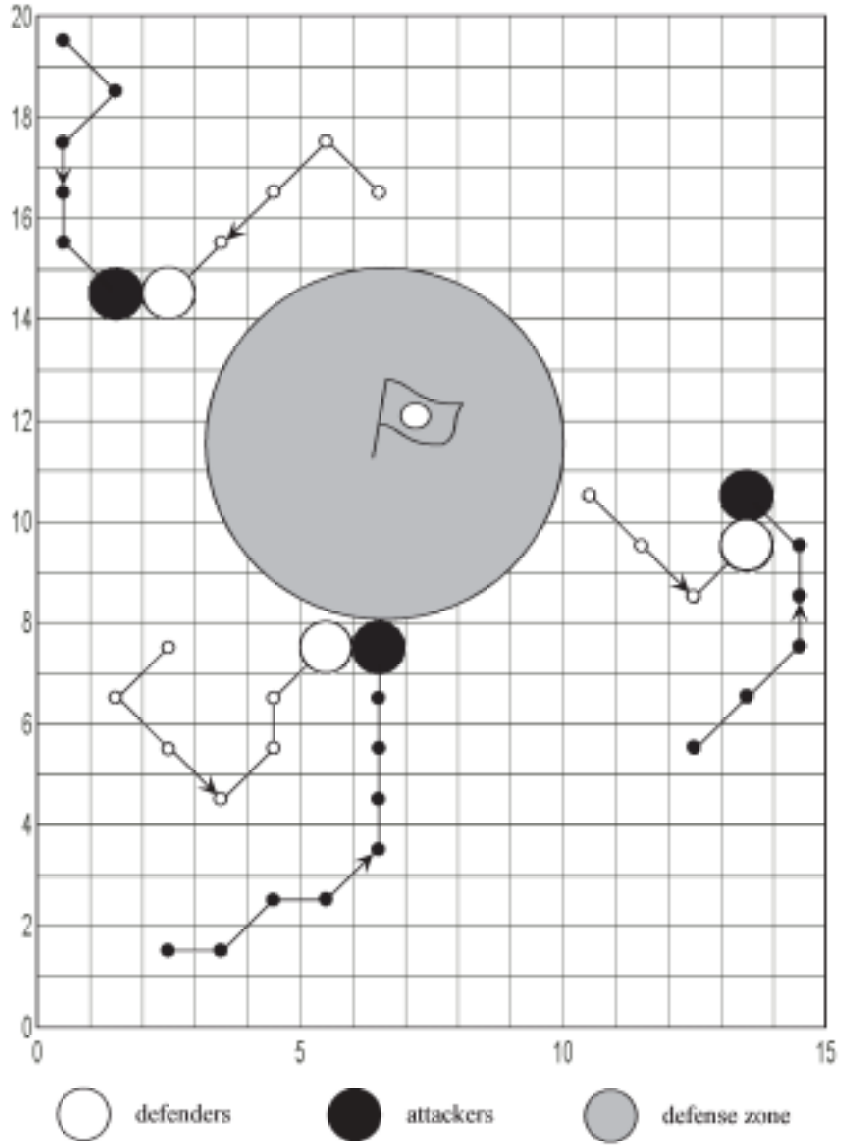


Figure 5.2: A control algorithm for defense with $N_p = 6$ is applied in the RoboFlag simulator created in Matlab. A stochastic feedback matrix for defense with $\bar{n}^a_{dsn} = 6$ is used. The attackers follow pre-specified paths unknown to the defenders.

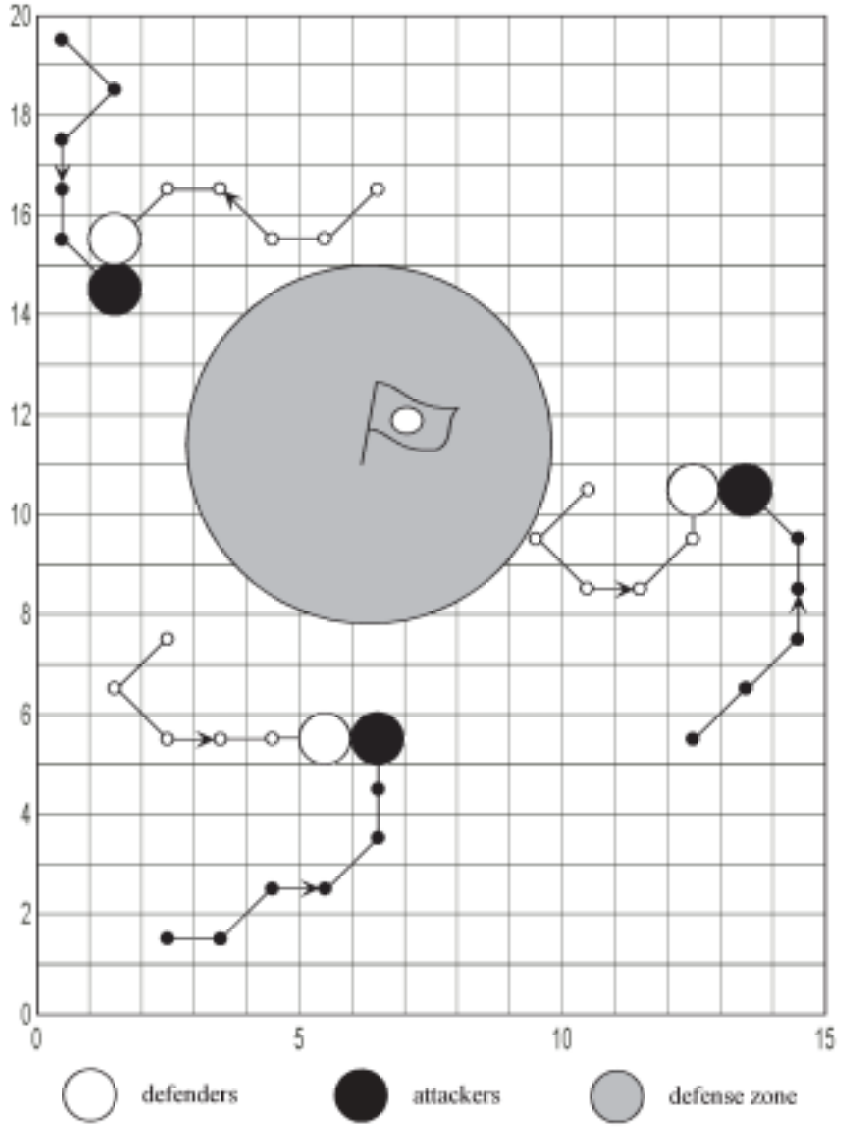


Figure 5.3: A control algorithm for defense with $N_p = 6$ is applied in the "Cornell RoboFlag Simulator". A stochastic feedback matrix for defense with $\bar{n}^a_{dsn} = 6$ is used. The attackers follow pre-specified paths unknown to the defenders.

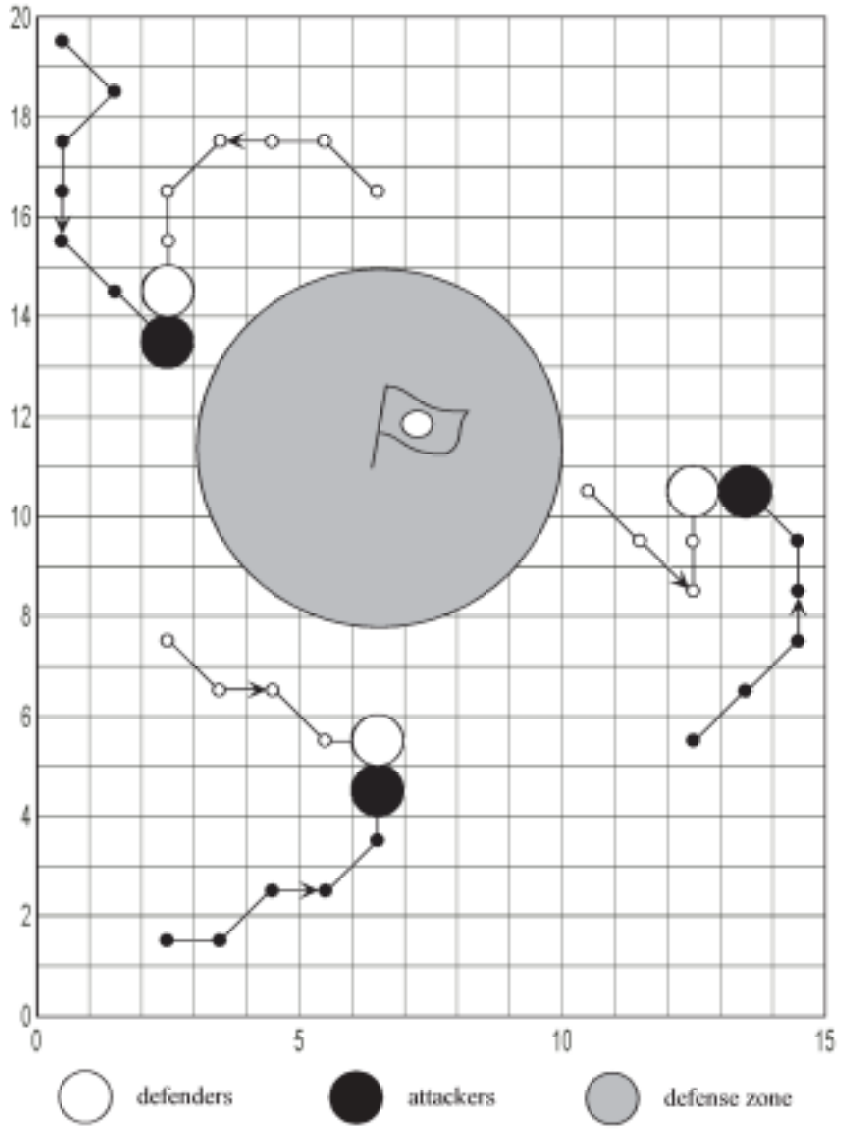


Figure 5.4: A control algorithm for defense with $N_p = 6$ is applied in the RoboFlag simulator created in Matlab. A stochastic feedback matrix for defense with $\bar{n}^a_{dsn} = 6$ combined with a velocity-based feedback matrix is used. The attackers follow pre-specified paths unknown to the defenders.

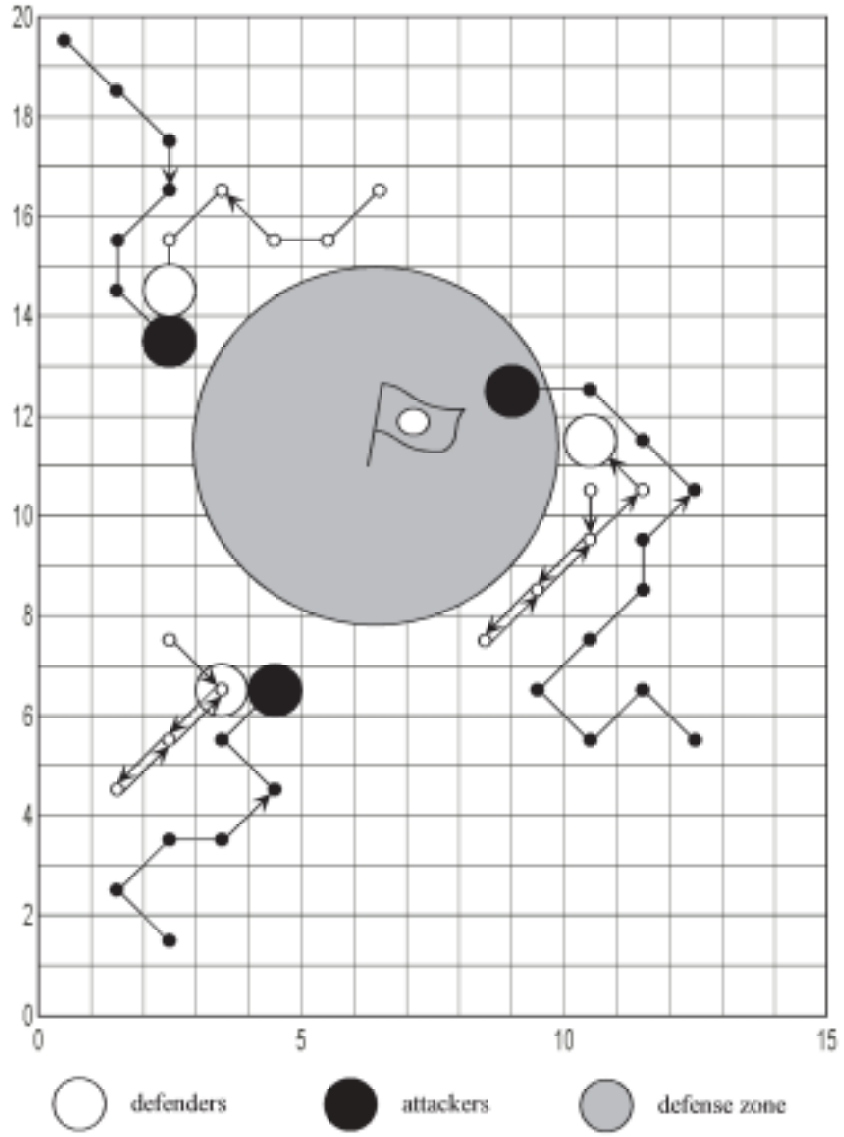


Figure 5.5: A control algorithm for defense with $N_p = 6$ is applied in the RoboFlag simulator created in Matlab. The defenders implement a stochastic feedback matrix for defense with $\bar{n}^a_{dsn} = 6$. The attackers use a control algorithm for attack with $N_p = 6$, $w^a_{dz} = 0.01$ and $w^a_d = 0.99$ in combination with a stochastic feedback matrix for attack.

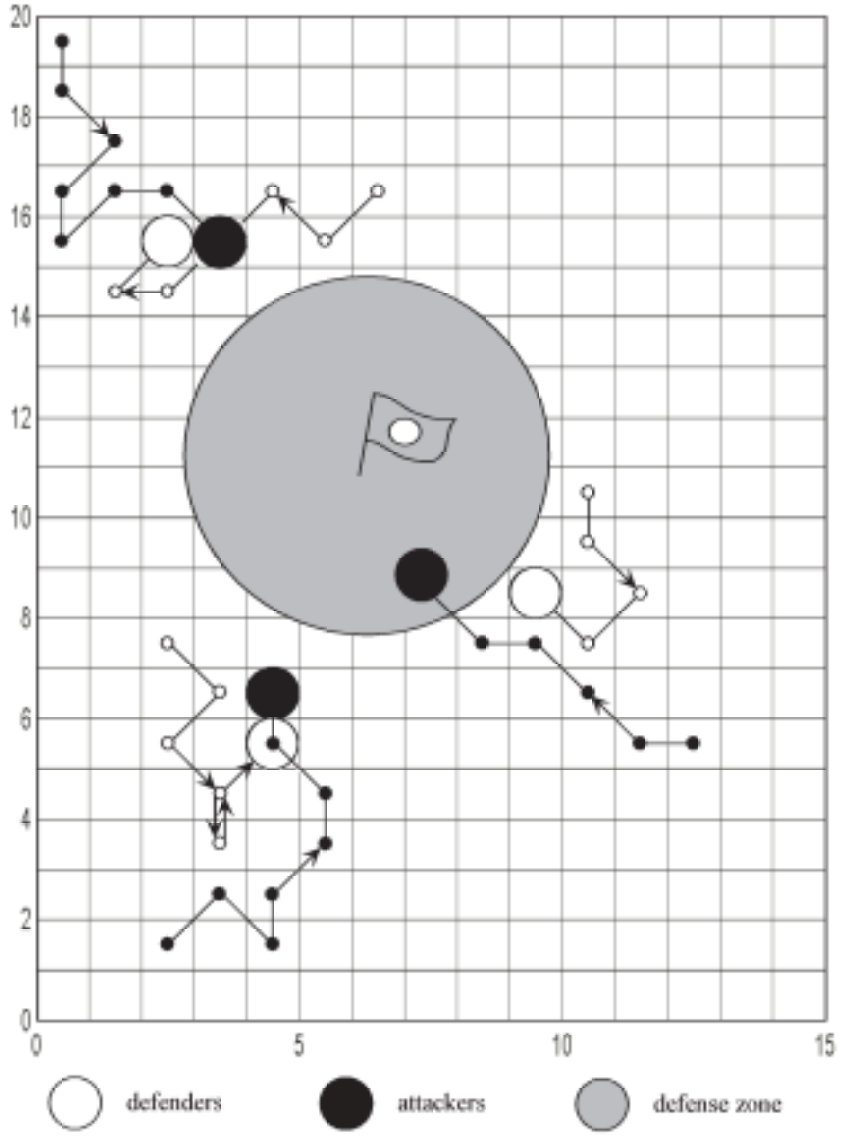


Figure 5.6: A control algorithm for defense with $N_p = 6$ is applied in the RoboFlag simulator created in Matlab. The defenders implement a stochastic feedback matrix for defense with $\bar{n}^a_{dsn} = 6$. The attackers use a control algorithm for attack with $w^a_{dz} = 0.01$ and $w^a_d = 0.99$ only at the first and last stage of the optimization horizon, otherwise $w^a_{dz} = 0$ and $w^a_d = 1$. The attackers also implement a stochastic feedback matrix for attack.

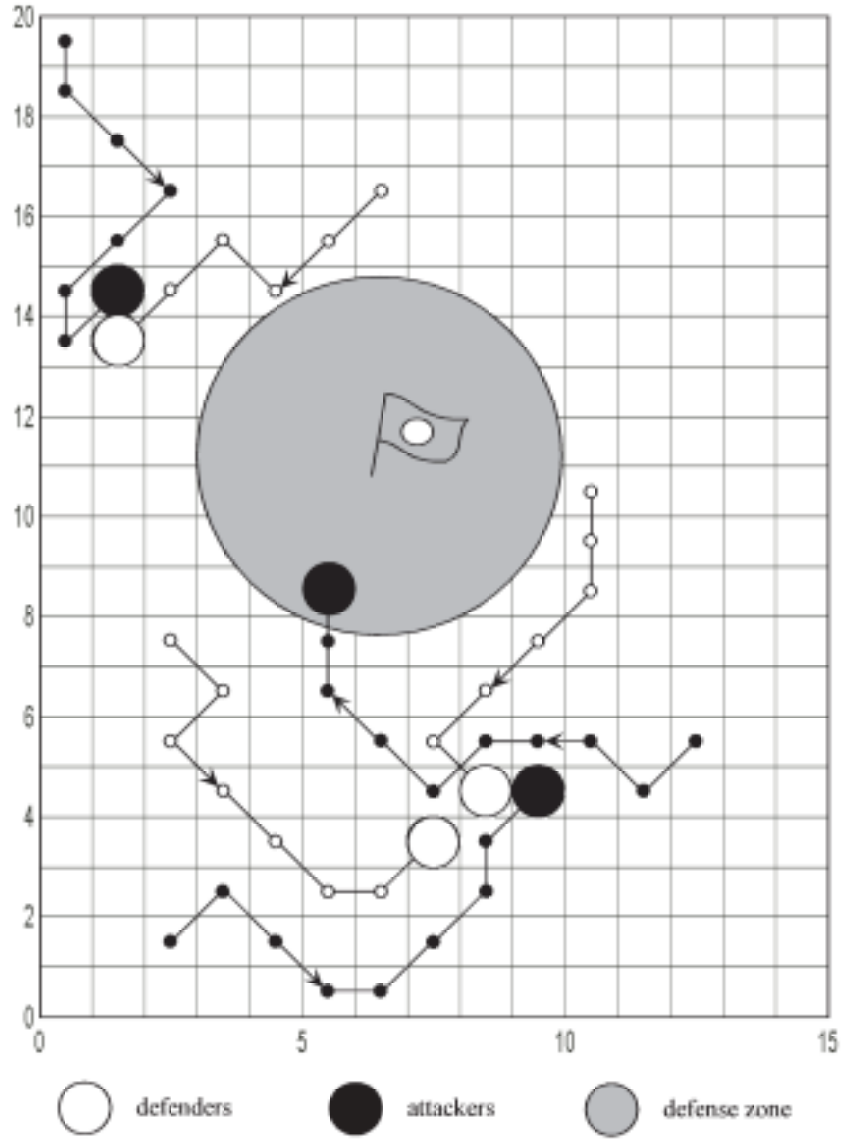


Figure 5.7: A control algorithm for defense with $N_p = 6$ is applied in the RoboFlag simulator created in Matlab. The defenders use the objective function of Section 4.3.8 with weights $w_a^d = 0.95$ and $w_{lz}^d = 0.05$ and a stochastic feedback matrix for defense with $\bar{n}_{dsn}^a = 6$. The attackers use a control algorithm for attack with $w_{dz}^a = 0.01$ and $w_d^a = 0.99$ only at the first and last stage of the optimization horizon, otherwise $w_{dz}^a = 0$ and $w_d^a = 1$. The attackers also use a stochastic feedback matrix for attack.

CHAPTER 6

Conclusions and Future Work

This thesis was a small contribution to the problem of multi-vehicle path planning in an adversarial environment. Our objective was to find an optimization method that will provide both online inclusion of the environment's uncertainties and computational efficiency which is still an open issue. To this end, we explored the utility of linear programming for trajectory planning in multi-vehicle systems with adversaries.

More specifically, we first constructed a model that describes the evolution of several resources in an arena of sectors. This model is a large-scale linear flow subject to various positivity constraints and can be used in describing situations where friendly and enemy resources are engaged in an arena of sectors. The reason for this choice is grounded on the linearity of this model that allows for using linear objective functions in an optimization problem for friendly planning.

Although enemy resources can be viewed as a disturbance to the system of friendly resources, they can be modelled as state dependent. Based on that we derived a linear-programming-based planning for friendly resources allocation. Moreover, this method was grounded on the assumption that the adversaries follow a stochastic feedback matrix which can describe their possible future attitude and is included in the optimization.

Then, we explored the utility of this linear-programming-based planning for resources allocation in deriving optimal paths for multi-vehicle control systems with adversaries. In particular, we considered the RoboFlag competition which involves two teams of robots with opposing interests, the defenders and the attackers. We

derived control algorithms for both defense and attack that are based on the linear-programming-based planning for resources allocation. In this case, both defenders and attackers are modelled as different resource types in an arena of sectors. Moreover, since adversaries are modelled as state-dependent, various stochastic feedback laws based on their current position and/or velocity could describe their possible future attitude, which is included in the optimization.

Finally, we tested these algorithms for either defense or attack in the RoboFlag simulator. In particular, the control algorithm for defense in combination with a stochastic feedback matrix for defense and/or a velocity-based stochastic feedback matrix gives satisfactory results against "smart" attackers, while at the same time was computationally efficient. On the other hand, the control algorithm for attack guarantees that a number of attackers infiltrate the defense zone independently of the defenders' path planning.

Although these algorithms were tested only in the RoboFlag simulator, it can also be used in deriving efficient paths in other multi-vehicle tasks. In particular, the objective functions used in these optimizations can be easily modified to include any other obstacle avoidance tasks and in different environments. Moreover, any possible uncertainty of the environment can be modelled using stochastic feedback matrices which can be easily included in the optimization problem. In other words, the proposed path planning could be a tool for deriving efficient paths in a great variety of future multi-vehicle tasks.

References

[1] Alur, R., A. Das, J. Esposito, R. Fierro, Y. Hur, G. Grudic, V. Kumar, I. Lee, J. P. Ostrowski, G. Pappas, J. Southall, J. Spletzer, and C. Taylor, "A framework and architecture for multirobot coordination," *Experimental Robotics: LNCS Series*, Springer-Verlag, 2001.

[2] Bemporad, A. and M. Morari, "Robust model predictive control: A survey," in A. Garully, A. Tesi and A. Vicino, editors, *Robustness in Identification and Control*, vol. 245. Springer-Verlag Lecture Notes in Control and Information Sciences, 1999.

[3] Bemporad, A. and M. Morari, "Control of systems integrating logic, dynamics and constraints," *Automatica*, Vol. 35, March 1999, pp. 407-428.

[4] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization*, Athena Scientific, Belmont, Massachusetts, 1997.

[5] Bortoff, S., "Path Planning for UAVs," *Proceedings of the American Control Conference*, Chicago, IL USA, June 2000, pp. 364-368.

[6] Bowling, M. and M. Veloso, "Multiagent learning using a variable learning rate," *Artificial Intelligence*, Vol. 136, 2002, pp. 215–250.

[7] Chandler, P. R. and M. Pachter, "Research Issues in Autonomous Control of Tactical UAVs," *Proceedings of the American Control Conference*, Philadelphia, PA USA, June 1998, pp. 394-398.

[8] Chandler, P. R., S. Rasmussen and M. Pachter, "UAV Cooperative Path Planning," *In Proceedings of AIAA Guidance, Navigation and Control Conference*, Denver, Colorado USA.

[9] Chandler, P. R., "UAV Cooperative Control," *Proceedings of the American Control Conference*, Arlington, VA USA, June 25-27, 2001, pp. 50-55.

[10] Dahleh, M. A., and I. J. Diaz-Bobillo, *Control of uncertain systems: A linear programming approach*, Prentice-Hall, Inc., Englewood Cliffs, NJ USA, 1995.

[11] D'Andrea, R. and M. Babish, "The RoboFlag Testbed," *Proceedings of the American Control Conference*, Denver, CO USA, June 4-6, 2003, pp. 656-660.

[12] D'Andrea, R. and R. M. Murray, "The RoboFlag Competition," *Proceedings of the American Control Conference*, Denver, Colorado USA, June 4-6, 2003, pp. 650-655.

[13] Daniel-Berhe, S., M. Ait-Rami, J. S. Shamma, J. L. Speyer, "Optimization Battle Management," *Proceedings of IEEE - American Control Conference 2001, ACC' 2001*, Arlington, VA USA, June 25-27, 2001, pp. 4711-4715.

[14] Dogan, A., "Probabilistic Path Planning for UAVs," *American Institute of Aeronautics and Astronautics, Inc.*, 2003.

[15] Dunbar, B. W. and R. M. Murray, "Model Predictive Control of Coordinated Multi-Vehicle Formations," *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada USA, December 2002, pp. 4631-4636.

[16] Earl, M. G. and R. D'Andrea, "A study in cooperative control: The RoboFlag Drill," *Proceedings of the American Control Conference*, Anchorage, AK USA, May 8-10, 2002, pp. 1811-1812.

[17] Earl, M. G. and R. D'Andrea, "Modeling and control of a multi-agent system using mixed integer linear-programming," *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, Nevada USA, December 2002, pp. 107-111.

[18] Flint, M., M. Polycarpou, E. Fernandez-Gaucherand, "Cooperative path-planning for autonomous vehicles using dynamic programming," *15th Triennial World Congress*, Barcelona, Spain.

[19] Harmon, M. E., and L. C. Baird, "Multi-agent Residual Advantage Learning with General Function Approximation," Wright Laboratory report WL-TR-96-1065, Wright-Patterson AFB, OH 45433.

[20] ILOG, Inc. CPLEX 7.1, http://www.ilog.com/products/cplex.

[21] Jun, M. and R. D'Andrea, "Path planning for unmanned aerial vehicles in uncertain and adversarial environments," In the book chapter of "Cooperative Control: Models, Applications and Algorithms," Kluwer, edited by S. Butenko, R. Murphey and P. Pardalos.

[22] Kailath, T., *Linear Systems*, Prentice Hall, Inc., 1980.

[23] Kott, A., "A proposal for an operational-level game: Seady storm," 1999.

[24] Latombe, J.C., *Theory of Robot Control* , Kluwer Academic Publishers, Boston, MA USA, 1991.

[25] Mataric, M. J., "Learning in Behavior-Based Multi-Robot Systems: Policies, Models, and Other Agents," *Cognitive Systems Research, special issue on Multi-disciplinary studies of multi-agent learning*, Ron Sun, ed., 2(1), April 2001, pp. 81-93.

[26] McLain, T.W., P.R. Chandler and M. Pachter, "A Decomposition Strategy for Optimal Coordination of Unmanned Air Vehicles," *Proceedings of the American Control Conference*, Chicago, IL USA, June 2000, pp. 369-373.

[27] McLain, T.W. and R.W. Beard, "Cooperative Path Planning for Timing-Critical Missions," *Proceedings of the American Control Conference*, Denver, CO USA, June 4-6, 2003, pp. 296-301.

[28] Ögren, P., "Formations and Obstacle Avoidance in Mobile Robot Control," *Doctoral Thesis*, Department of Mathematics, Royal Institute of Technology, Stockholm, Sweden, 2003.

[29] Pachter, M. and P. R. Chandler, "Challenges of Autonomous Control," *Control Systems Magazine, IEEE*, Vol. 18, No. 4, August 1998, pp. 92-97.

[30] Richards, A., J. How, T. Schouwenaars and E. Feron, "Plume Avoidance Maneuver Planning Using Mixed Integer Linear Programming," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Montreal, Canada, August 6-9, 2001.

[31] Richards, A., J. Bellingham, Michael Tillerson and J. How, "Coordination and control of multiple UAVs," AIAA Guidance, Navigation and Control Conference and Exhibit, Monterey, CA, 5-8 August, 2002.

[32] Richards, A. and J. P. How, "Aircraft Trajectory Planning with Collision Avoidance Using Mixed Integer Linear Programming," *Proceedings of the American Control Conference*, Anchorage, AK USA, May 8-10, 2002, pp. 1936-1941.

[33] Cornell RoboFlag web page: http://roboflag.mae.cornell.edu/

[34] Schouwenaars, T., B. De Moor, E. Feron, J. How, "Mixed integer programming for multi-vehicle path planning," *European Control Conference*, Porto, Portugal, 2001.