USING MULTIPLAYER DIFFERENTIAL GAME THEORY TO DERIVE EFFICIENT PURSUIT-EVASION STRATEGIES FOR UNMANNED AERIAL VEHICLES

A Thesis
Presented to
The Academic Faculty

by

Johan M. Reimann

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Electrical and Computer Engineering
Georgia Institute of Technology
August, 2007

USING MULTIPLAYER DIFFERENTIAL GAME THEORY TO DERIVE EFFICIENT PURSUIT-EVASION STRATEGIES FOR UNMANNED AERIAL VEHICLES

Date Approved: May 8th, 2007

ACKNOWLEDGEMENTS

The work presented in this thesis was financially supported by the ARO's Stochastic Pursuit-Evasion Differential Game for Autonomous Vehicles project.

The work would not have been possible without the insight and support of my advisor Professor George Vachtsevanos whose guidance has been invaluable. I also like to thank the members of my committee, Professor Magnus Egerstedt, Professor Eric Verriest, and Professor Vincent Mooney whose comments and suggestions have significantly impacted the direction of this research.

Finally, I would like to thank several of my co-workers and friends for the very interesting discussions and insights: Dr. Ben Ludington, Phillip Jones, Dr. Graham Drozeski, Dr. Liang Tang, Dr. Alex Liberson, and Dr. Jianhua Ge.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Mathematical Symbols

C^1	continuously differentiable function
dW	m-dimensional Brownian motion
$\varphi(\cdot)$	termination state cost
$f(\cdot)$	the vehicle dynamics
θ_{ej}	heading of evader *j*
θ_{pi}	heading of pursuer *i*
$H(\cdot)$	the Hamiltonian
$J(\cdot)$	the cost function
$\Psi(\cdot)$	termination condition
$L(\cdot)$	the integrated cost
$\vec{\lambda}$	the costate
P	second order costate
$\mathbb{R}^n$	*n* dimensional field of real numbers
t_0	initial time
T	termination time
$\vec{u}_e$	the evaders' control
$\vec{u}_p$	the pursuers' control
U_e	the set containing the evaders possible controls
U_p	the set containing the pursuers possible controls
$V(\cdot)$	the value of the game
$\vec{x}$	the complete state of the game

x_{pi}	*x* position of pursuer *i*
x_{ej}	*x* position of evader *j*
y_{pi}	*y* position of pursuer *i*
y_{ej}	*y* position of evader *j*

Acronyms and Abbreviations

DPRR	dynamic performance based region of responsibility
UAV	Unmanned Aerial Vehicle
HJBI	Hamilton-Jacobi-Bellman-Isaacs

SUMMARY

In recent years, Unmanned Aerial Vehicles (UAVs) have been used extensively in military conflict situations to execute intelligence, surveillance and reconnaissance missions. However, most of the current UAV platforms have limited collaborative capabilities, and consequently they must be controlled individually by operators on the ground. The purpose of the research presented in this thesis is to derive algorithms that can enable multiple UAVs to reason about the movements of multiple ground targets and autonomously coordinate their efforts in real-time to ensure that the targets do not escape. By improving the autonomy of multivehicle systems, the workload placed on the command and control operators is reduced significantly.

To derive effective adversarial control algorithms, the adversarial scenario is modeled as a multiplayer differential game. However, due to the inherent computational complexity of multiplayer differential games, three less computationally demanding differential pursuit-evasion game-based algorithms are presented. The purpose of the algorithms is to quickly derive interception strategies for a team of autonomous vehicles. The algorithms are applicable to scenarios with different base assumptions, that is, the three algorithms are meant to complement one another by addressing different types of adversarial problems.

The first algorithm, the *two-player decomposition approach*, reduces the larger problem of having multiple UAVs intercept multiple targets, into multiple two-player differential games. The Hamilton-Jacobi-Bellman-Isaacs (HJBI) equation is solved for each of the two-player games, and, based on the solution of each of the two-player games, each individual UAV is assigned a role of either intercepting a target directly or attempting to contain a target. This algorithm is applicable to scenarios in which the evading ground targets are slower and less agile than the pursuing vehicles.

The second algorithm presented utilizes the *maximum principle approach* to derive effective interception strategies. This approach solves the multiplayer differential games for situations where the size of the state space becomes too large to rely on the HJBI framework; however, additional smoothness requirements of the cost, constraint, and control functions are imposed on the problem formulation to utilize this approach.

Finally, the third algorithm presented, the *minimum-time decomposition approach*, reduces the general multiplayer problem into several minimum-time problems. It is in many ways similar to the two-player decomposition approach; however, as opposed to the two-player decomposition approach, the minimum-time decomposition approach is applicable in situations where the evading targets are faster or more agile than the pursuers.

The work presented in this thesis contributes to the field by providing:

- Fast adversarial decision algorithms which enable multiple cooperating vehicles to engage multiple targets efficiently by,
    - Decomposing the complete multivehicle problem into smaller and more manageable two-player or minimum time problems.
    - Effectively assign tasks to the individual pursuing vehicles based on the derived performance estimates.
- An effective approach to managing multiple unmanned systems in collaborative engagement scenarios.

Several simulations were conducted to verify the performance of the suggested approaches.

CHAPTER 1

INTRODUCTION AND MOTIVATION

In recent years, many military conflicts have taken place in an urban setting, and with increased experience in such conflicts, a demand for improving the existing military systems has emerged [1]. One of the main difficulties encountered in the urban warfare environment is the demand for rapid adaptation to changes in the environment. To facilitate a rapid response to a hostile move by an opposing force, it is necessary to provide the ground personnel with accurate and as close to real-time information as possible about the position and capabilities of the enemy. To address this need for real-time intelligence, Unmanned Aerial Vehicles (UAVs) have been used to provide video surveillance. The control algorithms that are currently guiding these vehicles requires significant human supervision [2], and if there are multiple vehicles in the same sector, the workload placed on the command and control personnel is immense. Hence, the Department of Defense (DoD) released a roadmap in 2005 that outlines what capabilities should be incorporated into the UAV platforms in the near future [3]. As shown in Figure 1, it is the DoD's goal to have fully coordinated swarms of UAVs by the year 2012.

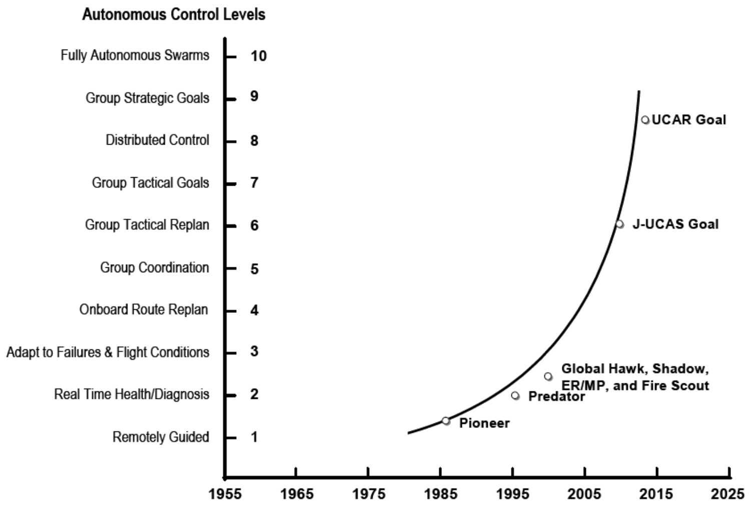


Figure 1. Trend in UAV autonomy.

To reach this goal, the Air Force, Army, Navy and DARPA's UAV research budget presented by the President for the FY05-09 totaled 1,662 million dollars distributed as shown in Figure 2.

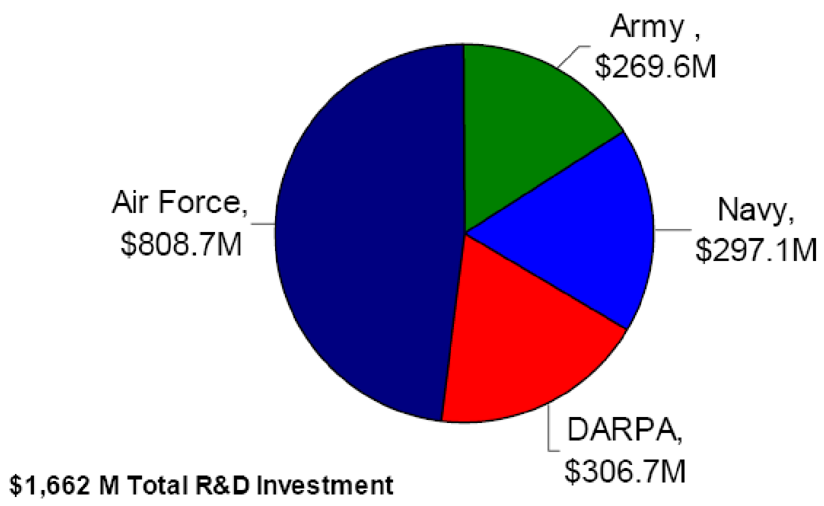


Figure 2. Investment in Research and Development of UAV Systems for FY05-09.

The research presented in the following chapters addresses the demand for increased autonomy of UAVs operating in urban environments. In particular, the problem of enabling a team of UAVs to make fast and effective decisions when faced with an intelligent opponent is analyzed and viable control algorithms are presented. By improving the autonomy of the UAVs, it is possible for the team of UAVs to rapidly adapt to scenario changes with limited supervision by a human operator.

A typical urban warfare scenario is depicted in Figure 3. Two UAVs are tasked with surveying an area for potential targets, and, upon detection of potential targets, classifying and tracking them. In this scenario, the targets are aware of the UAVs, therefore they will execute maneuvers that make interception as difficult as possible. It is the UAVs' objective to intercept the targets as fast as possible so that Command and Control is provided with timely and detailed information about enemy movements.

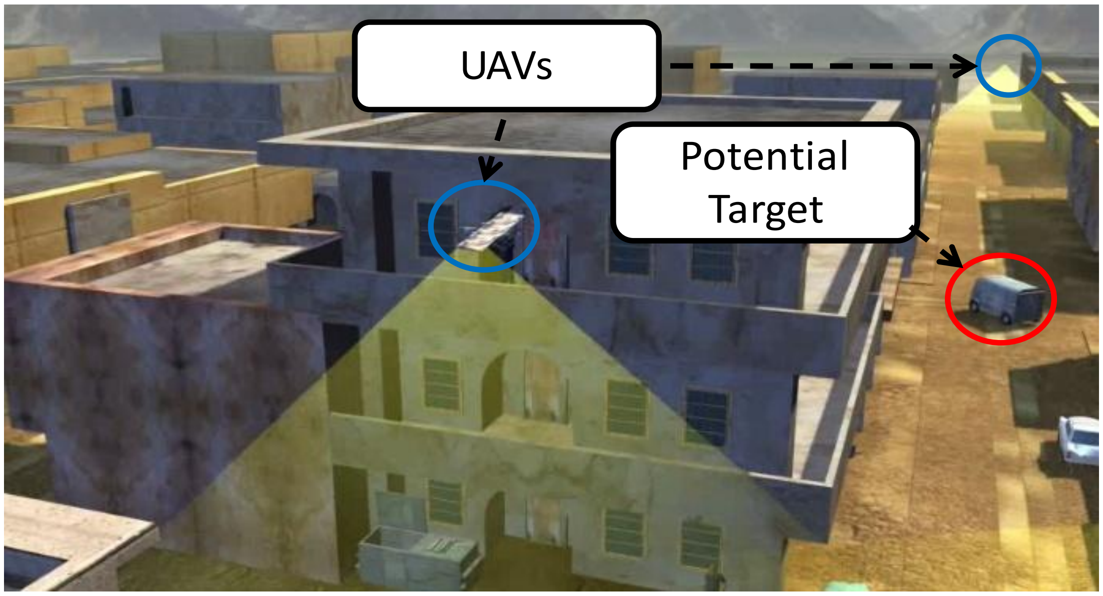


Figure 3. Two fixed wing UAVs are surveying an urban environment for potential targets.

When dealing with multiple pursuing UAVs and multiple evading ground targets, determining effective adversarial strategies is exceedingly difficult. The main difficulty encountered when solving such problems is the vast number of possible actions that have to be considered and the limited time available to make a decision. To further complicate matters, the movements of the targets and the UAVs are constrained by the vehicles' dynamics and the obstacles inherently present in an urban environment.

1.1 Overview of the Approach

To address these problems, several multiplayer differential game theory based approaches to adversarial reasoning are presented. Game theory has for a number of years been used to solve problems in economics and applied mathematics. These problems consist of multiple decision makers that must interact with each other in an attempt to influence the outcome of a given situation to their advantage. To be a decision maker or a *player* two criteria must be met. First, the player must be able to influence the outcome of the situation. If the player is unable to influence the outcome, the player is simply not a part of the game and can therefore be eliminated from the problem description. Second, the player must have a vested interest in the outcome of the situation, since if the player is indifferent to the outcome, the influence that player may be able to exercise over the system can simply be modeled as a noisy system disturbance.

Since the actions of UAVs are commonly described by a set of differential equations, differential game theory can be utilized to guide the UAVs in adversarial situations. Differential game theory is a branch of game theory dealing with problems in which the interaction between the players is a continuous process described by a set of differential equations. Hence, the players in a differential game must be able to analyze the outcome of a continuously evolving problem if they are to manipulate the situation to their advantage.

Since multiplayer differential games are very complex to solve even for very simple scenarios, many researchers have had difficulties solving differential games quickly [4]. The inherent complexity of differential games can easily be visualized by considering the classical traveling salesperson problem. In the traveling salesperson problem, it is the objective of a salesperson to visit a number of cities while minimizing the total distance traveled. This problem is not solvable in polynomial time due to the rapid increase in the number of possibilities that must be considered as the number of cities increases [5], however researchers have suggested approximation techniques that can be solved in polynomial time [6]. Since the traveling salesperson problem essentially only has one player, it is not strictly speaking a game, so to pose the problem more like a differential game, let the salesperson's or pursuing player's movements be governed by a set of differential equations. Furthermore, imagine that the cities that have to be visited are mobile and their movements are also governed by a set of differential equations. Now the traveling salesperson problem is solidly placed in the realm of differential games. It is the cities' goal to evade the salesman, or equivalently make the salesperson travel as far as possible, while the traveling salesperson will attempt to visit all the cities while traveling the shortest distance possible. This new problem is still not solvable in polynomial time, and the addition of having to solve large systems of differential equations not only increases the algorithmic complexity but also imposes some theoretical limits concerning the type of problems that can be solved.

In essence, multiplayer differential games are in most cases very difficult to solve due to the vast number of combinations that have to be considered. Therefore, it is important to limit the number of possibilities before attempting to solve a differential game. The work presented here describes three effective ways to limit the space over which the differential games must be solved. First, a decomposition based technique is presented. The basic idea behind the approach is to reduce the general multiplayer game into several smaller and easier to solve two-player differential games, and based on the estimated interception times obtained from each of the smaller games, assign pursuers to each of the evading targets. To draw a parallel to the traveling salesperson problem, the decomposition approach would solve the problem by calculating the estimated distance to each of the cities, and then make the salesperson travel to the closest one. It is well established that such a greedy-type approach is not optimal for the traveling salesperson problem, and as such the decomposition approach will also provide suboptimal solutions.

The second approach presented addresses a particular situation in which the two-player decomposition approach is guaranteed to fail. If the evading targets are much faster than the pursuers, the two-player games do not have a solution. Hence, the pursuers are required to leverage their numbers to arrive at a solution, that is, the prescribed capture conditions can only be reached if at least two players cooperate.

To effectively accomplish this cooperation between the pursuers without increasing the computational complexity, the multiplayer differential game is decomposed into several minimum-time problems. The resulting solutions to the minimum-time problems are then combined to a minimum-time map. Based on the map, effective containment points are determined, that is, the evading targets potential escape corridors are being closed off by the pursuers. The main idea behind this strategy is to reduce the worst possible case such that the targets are slowly being cornered.

The last method used to solve the multiplayer game is based on the Maximum Principle. The Maximum Principle is a powerful tool used to solve optimal control problems. Instead of solving an optimal control problem for an entire family of problems, the Maximum Principle utilizes not only the prescribed termination conditions, but also the given initial conditions. By using all of this information it is possible to derive a single optimal trajectory. Hence, the overall computational complexity can be reduced significantly since the dimensionality of the problem is reduced. However, the technique is only appropriate for certain problems. For more detailed information about the Maximum Principle consult appendix A.

As a consequence of limiting the number of possibilities considered before attempting to solve the problem, the solutions are not guaranteed to be optimal. The challenge is then to show that the derived solutions are effective even though they are not optimal. Also, since the three approaches are meant to address three different types of adversarial problems, there are certain scenarios in which more than one of the techniques can solve the problem. However, given a particular conflict scenario, additional constraints such as available computational resources on board the UAVs or communication limitations may make one of the algorithms a better choice than the others.

1.2 General Assumptions

When deriving effective adversarial strategies, it is important for the members of the coalitions to not only know the state of the other players in the game but also know their capabilities. Hence, throughout this thesis it is assumed that all the players have access to the state information of the other players, and the dynamic constraints that each of the players must adhere to are also known. For differential games in which these assumptions hold are generally called complete information games.

For the complete information requirement to be valid for the current UAV platforms, additional modules must be incorporated into their systems. In Figure 4, some of the required modules are depicted. The implementation of these modules is challenging, and image processing, target tracking, and model building are currently topics of intense research.

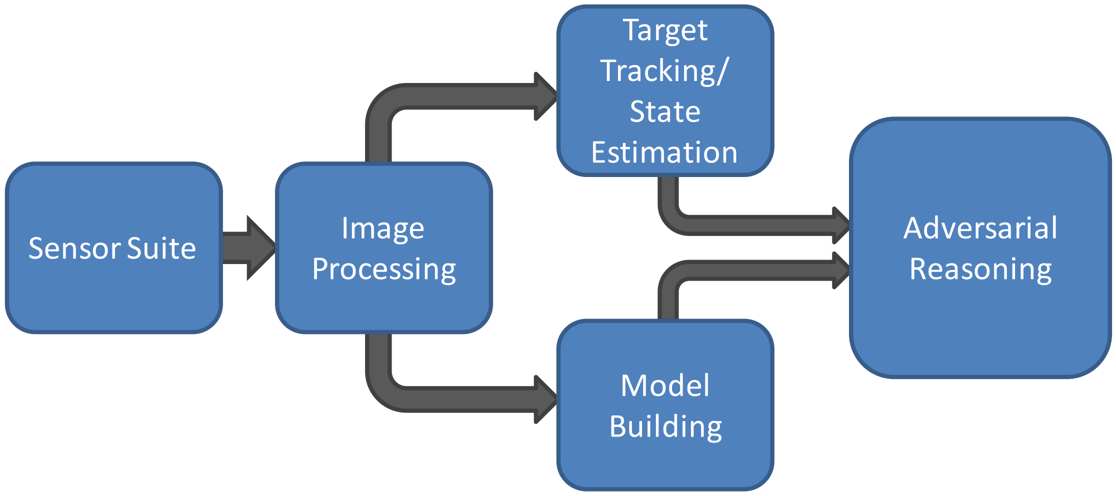


Figure 4. Several algorithms must be used to extract the pertinent information from the sensor suite.

In addition to the complete information assumption, it is also assumed that there are only two coalitions involved in the adversarial scenarios. A coalition is a group of players or decision makers that have similar objectives and they are therefore combining their resources to achieve the best possible outcome. This assumption simplifies the problems considered significantly, and even though the techniques described in the following chapters may be extended to cover situations with multiple coalitions, such situations are considered to be outside the scope of the work presented.

1.3 Organization of the Thesis

To effectively convey the details of the three approaches, much of the more technical material is placed in appendices. In addition, the first couple of appendices are dedicated to background information about the Hamilton-Jacobi-Bellman (HJB) equation, the Maximum Principle and numerical solution techniques commonly used to solve the resulting differential equations. Some background information along with some current techniques used to solve adversarial problems is provided in chapter 2. Chapter 3, 4 and 5 are each dedicated to the three suggested approaches to solve differential games in real-time, and chapter 6 covers additional implementation considerations such as grid construction and refinement. Finally, in chapter 7 some concluding remarks and possible future extensions of the work are presented.

CHAPTER 2

BACKGROUND

2.1 Multivehicle Control Techniques

Many researchers have studied the problem of effectively controlling swarms of robots in an attempt to utilize the collaborative benefits inherently present when multiple robots are in the vicinity of one another. One of the most commonly implemented architectures used to solve this problem is derived from observations made in nature. When a flock of birds migrates south for the winter, they do not have an agreed upon travel plan that every bird is following. Instead, each bird relies on its own senses to guide its movements in the flock, and ultimately the energy consumed by each individual bird in the flock is minimized by constructing a formation with the other birds. By emulating this flocking behavior in robotic swarms, it is possible to implement distributed guidance laws that effectively enable swarms of robots to collectively achieve goals at a minimum cost or perhaps even achieve goals that were otherwise unobtainable.

A good overview of such systems can be found in [7, 8]. In [7] a thorough discussion of flocking algorithms is provided. Several interesting questions such as convergence and stability of flocks are covered. In addition, the problem of safely allowing a flock of autonomous vehicles to circumnavigate obstacles without causing collisions is discussed. However, in many cases the swarm of robots has to perform several different tasks, and consequently a decentralized approach to coordination that allows each member of the swarm to perform different actions must be adopted. The swarm is able to tackle more diverse problems effectively by providing each robot with a different skill set.

Such approaches often rely on behavior type control schemes in which each individual member of the swarm is assigned a particular role, and if it executes this role well, the entire flock will benefit from the success of the individual [9]. Such systems have been implemented on RoboCup teams [10, 11] in which a team of autonomous robots play a game of soccer against an opposing team of robots. Since the robots in RoboCup do not have global situation awareness, the cooperative algorithms should not only handle soccer related tasks such as passing or blocking the ball, but also enable the players to explore the environment to determine the state of the other players in the game. In addition, the robots have limited computational resources onboard, which effectively exclude them from utilizing very complex planning algorithms. Consequently, low complexity ad-hoc assignment algorithms are commonly used to guide the robots during the soccer game. In [9], the complexity of some task assignment algorithms (ALLIANCE, BLE, M+, MURDOCH, First-price auction, Dynamic role assignment) is listed and similarities between the algorithms are highlighted. In [12], an interval programming approach is used to assign behaviors to an autonomous vehicle. The purpose of the interval programming approach is to effectively determine the best possible behavior given a set of competing interval programming functions and an overall objective function. The approach was demonstrated on autonomous marine vehicles. In [13], the dual problem of avoiding an obstacle was investigated, and a reactive navigation system was investigated and implemented on the MAGELLAN PRO platform from IRobot.

Determining the effective behaviors for the team of robots can be difficult. In many applications of the behavior based approach to team coordination, it is the researchers’ task to determine a set of effective behaviors that will work in most situations. However, to make the control system more robust, it may be beneficial to have the robots derive their own strategies online. Such systems have been developed for simple adversarial problems consisting of only two players. Concepts from differential game and optimal control theory have been used to derive optimal control strategies online and the resulting guidance laws have been used to develop effective missile control systems and to guide Unmanned Aerial Vehicles in adversarial situations [14, 15, 16]. The differential game problem can be described as follows: A missile must intercept a moving target as fast as possible. The target is aware of the present danger and will attempt to stay alive for as long as possible by applying an evasive strategy. The problem is then to determine the optimal interception and evasion strategies.

R. Isaacs posed this question and presented a closed form solution for a given set of dynamic equations by applying two-player pursuit-evasion differential game theory [17]. However, since missile dynamics are significantly different from the dynamics used by R. Isaacs in his analysis, the problem is still being studied today. More recently, effective interception strategies for problems with quadratic cost functions have been found by applying extended linearization techniques and solving the resulting Riccati equations [14]. Other examples of determining optimal strategies for differential games consisting of only two players are discussed in [18, 19].

For situations that do not easily lend themselves to linearization or for which the cost function is not quadratic, numerical techniques have been applied. In [4, 20, 21], collision avoidance strategies were found by applying a Gauss-Seidel [22] solution technique to the differential game problem. However, the computational complexity related to determining the optimal solution to multiplayer differential pursuit-evasion games is very high, since a global performance map must be constructed numerically for all the possible states of the game. Note that the resulting strategies will be optimal if a particular problem can be solved easily using multiplayer differential game theory.

To avoid the excessive computational burden related to solving multiplayer differential games, researchers have been investigating several different map building techniques. These techniques are very practical in nature, since the UAVs in the swarm collect information about the environment using their local sensor suite, and based on the information each member of the swarm provides, a global map is constructed. The map contains information about the obstacles present location and where the targets are most likely to be [23, 24]. The collaborative strategies employed by the UAVs rely heavily on the information captured in the map. Many different ad-hoc type strategies have been tested on these systems, and the performance of these systems appears promising; however, it is difficult to provide very accurate performance guarantees.

In an attempt to derive some performance guarantees, some researchers have decomposed a map the environment into smaller triangular regions, and then superimposed a tree structure on the map. Given this tree structure, it is possible to guarantee that all the targets can be captured in finite time using a random tree search [25, 26]. By superimposing a tree structure onto the space over which the game is being played, and then proving performance based on the tree structure, the proof will only be valid for as long as the tree structure is a good approximation of the environment. As shown in Figure 5, this assumption may not hold true in environments containing obstacles. Figure 5 A through C depicts the ideal case, where the tree structure is a valid approximation of the game environment, while Figure 5 D illustrates an example where a cycle is introduced by an obstacle in the environment, which invalidates the performance guarantees.

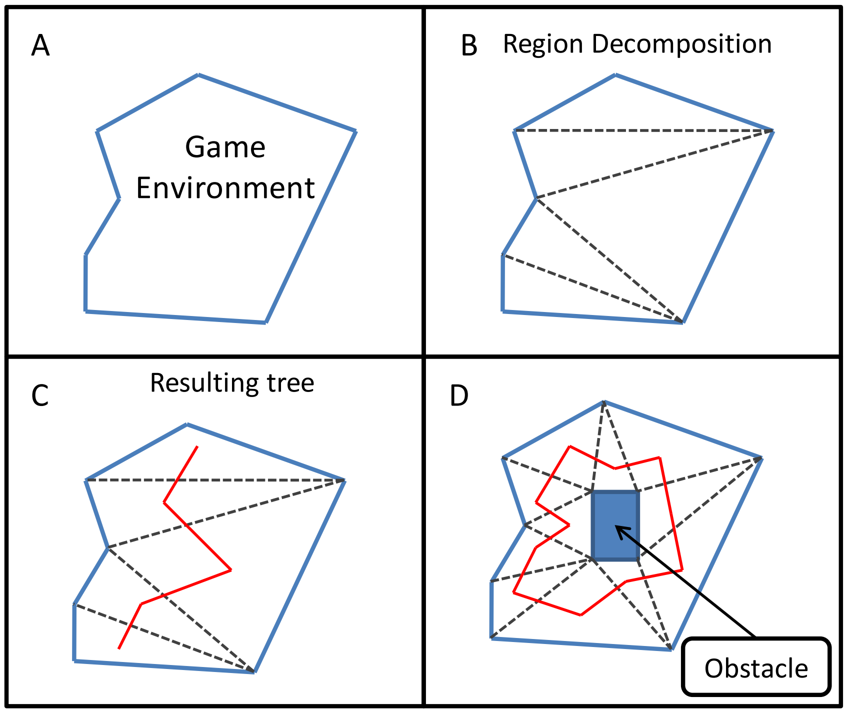


Figure 5. A) The game environment in which the pursuit-evasion game is played. B) The region is decomposed into triangular regions. C) A tree structure is imposed on the environment. D) A case in which the performance guarantees provided are invalidated by the presence of an obstacle.

The approaches presented in the following chapters are a mixture of the behavior type approaches and the differential game approaches. The underlying idea is to avoid prescribing a particular set of behaviors of which a potential opponent may take advantage. The individual behaviors are derived online by solving smaller and less complex differential games while the team objectives are assigned using matching techniques. The decoupling of the individual vehicles from one another ensures that the space over which the differential games are solved remains small. Consequently, it is possible to derive collaborative strategies fast and effectively.

2.2 Solving Differential Games

It is important to address the theoretical background to establish some of the limitations imposed by this framework. Strong ties exist between differential game theory and optimal control theory; therefore, appendix A, is dedicated to reviewing the Hamilton-

Jacobi-Bellman and the Maximum Principle frameworks along with the underlying assumptions made when deriving them.

The main difference between a standard optimal control problem and a differential game is related to the adversarial nature of differential games. In an optimal control problem, it is the objective to either maximize or minimize a cost function, while in a differential game a saddle point or equilibrium point is being sought by the two conflicting parties. Hence, the standard Hamilton-Jacobi-Bellman equation given in appendix A is modified to include a maximizing *and* a minimizing term. Let the system to be controlled be described by a set of differential equations of the form,

$$\dot{\vec{x}} = f(\vec{x}, \vec{u}_p, \vec{u}_e, t) \tag{1}$$

where $\vec{x}$ is the current state of the system, $\vec{u}_p$ is the control of the pursuing coalition, $\vec{u}_e$ is the control of the evading coalition and *t* is time. A coalition is, in this context, considered to be a group of players who collaborate to achieve a mutually beneficial goal.

Furthermore, let the objective function be given by,

$$\min_{\vec{u}_p} \max_{\vec{u}_e} J = \varphi\big(x(T)\big) + \int_{t_0}^{T} L\big(\vec{x}, \vec{u}_p, \vec{u}_e, t\big) d\tau, \tag{2}$$

where *L()* is the integral cost function, *φ()* is the final state cost, *T* is the termination time, and t_0 is the initial time. The cost function captures the essence of the differential game problem. Since the pursuing coalition is attempting to minimize the quantity that the evading coalition is attempting to maximize, they are forced to consider each others' control strategies. Note, that the differential game description could be generalized further by considering the problem involving more than two coalitions. In such situations, there could potentially be several cost functions describing the different points of contention between the coalitions with a maximum number of $n \cdot (n-1)/2$ cost functions, where n is the number of coalitions. However, in this work we will for simplicity only consider two coalitions, namely a pursuing and an evading coalition.

Given the dynamics and the cost function described by Equations 2 and 3, the Hamilton-Jacobi-Bellman-Isaacs equation is given by the following differential equation,

$$\begin{cases} \frac{\partial v}{\partial t} + min_{\vec{u}_p \in U_p} max_{\vec{u}_e \in U_e} \{ \langle \vec{\lambda}, f(\vec{x}, \vec{u}_p, \vec{u}_e, t) \rangle + L(\vec{x}, \vec{u}_p, \vec{u}_e, t) \} = 0 \\ v|_{t=T} = \varphi(\vec{x}(T)) \end{cases} \tag{3}$$

where $(t, \vec{x}, \vec{u}, \vec{\lambda}) \in [t_0, T] \times \mathbb{R}^n \times U \times \mathbb{R}^n$, $\vec{\lambda} = \left[\frac{\partial v}{\partial x_1}, \frac{\partial v}{\partial x_2}, \cdots, \frac{\partial v}{\partial x_n}\right]$, U_p and U_e are the set of admissible controls for the pursuers and the evaders respectively, and $\langle , \rangle$denotes the inner-product of $\vec{\lambda}$ and $f(\vec{x}, \vec{u}, t)$.

It is important to note the order in which the *min* and the *max* operations are performed, and how the ordering influences the outcome of the differential game.

Definition 1: The Value of a differential game at a particular state, $(\vec{x}, t) \in \mathbb{R}^n \times [t_0, T]$ is defined to be the optimal cost of reaching the target state from $(\vec{x}, t)$, that is,

$$V(\vec{x}, t) = \min_{\vec{u}_p} \max_{\vec{u}_e} J(\vec{x}, \vec{u}_p, \vec{u}_e, t) \tag{4}$$

Definition 2: Isaacs Condition: For any $\vec{\lambda} \in \mathbb{R}^n$and $(\vec{x}, t) \in \mathbb{R}^n \times [t_0, T]$, the equality

$$\begin{gathered} min_{\vec{u}_p} max_{\vec{u}_e} \{ \langle \vec{\lambda}, f(\vec{x}, \vec{u}_p, \vec{u}_e, t) \rangle + L(\vec{x}, \vec{u}_p, \vec{u}_e, t) \} = \\ max_{\vec{u}_e} min_{\vec{u}_p} \{ \langle \vec{\lambda}, f(\vec{x}, \vec{u}_p, \vec{u}_e, t) \rangle + L(\vec{x}, \vec{u}_p, \vec{u}_e, t) \} \end{gathered} \tag{5}$$

is valid.

Isaac's condition does not always hold, that is,

$$\begin{gathered} min_{\vec{u}_p} max_{\vec{u}_e} \{ \langle \vec{\lambda}, f(\vec{x}, \vec{u}_p, \vec{u}_e, t) \rangle + L(\vec{x}, \vec{u}_p, \vec{u}_e, t) \} \leq \\ max_{\vec{u}_e} min_{\vec{u}_p} \{ \langle \vec{\lambda}, f(\vec{x}, \vec{u}_p, \vec{u}_e, t) \rangle + L(\vec{x}, \vec{u}_p, \vec{u}_e, t) \} \end{gathered} \tag{6}$$

A depiction of such a situation is shown in Figure 6. In the case where the Isaac's condition does not hold, two different solutions must be considered; namely, the *upper value* and the *lower value*. The upper value is associated with the solution in which the evading target has a slight advantage over the pursuer while the lower value is associated with the pursuer being favored. The concept of upper and lower values is not always related to the notions of *super-* and *subsolutions* introduced by the viscosity solution approach, but in some problems they have been shown to be equivalent [27]. Viscosity solutions are used to find solutions to partial differential equations that are non-smooth. For a more detailed discussion of viscosity solutions consult appendix A and additional information about the general differential game framework can be found in [28].

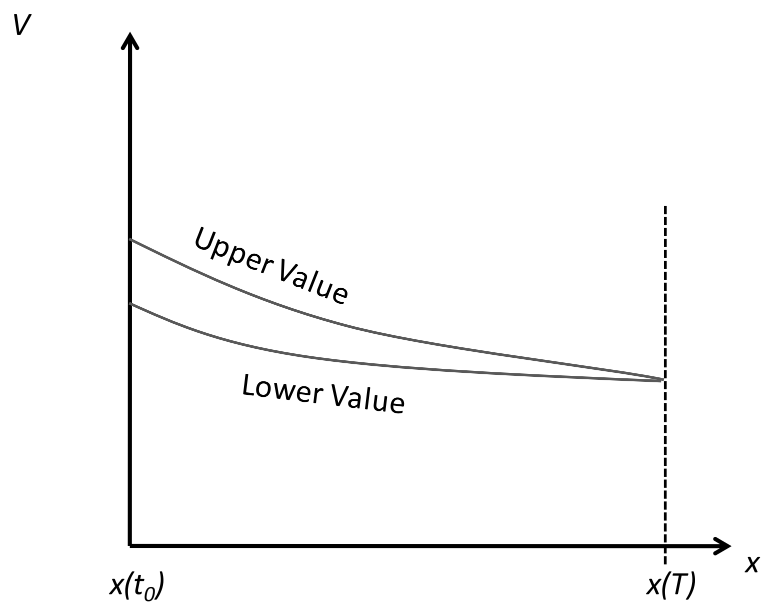


Figure 6. If Isaacs condition does not hold, the upper and lower value functions may diverge.

The solution the differential game problem described will generally speaking satisfy the Nash equilibrium strategy [29]:

Definition 3: Nash equilibrium strategy: Given the solution $F(\vec{u}_p^*, \vec{u}_e^*)$ where $\vec{u}_p^*, \vec{u}_e^*$ are the solution to the differential game problem, then $\forall \vec{u}_p \in U_p$ and $\vec{u}_e \in U_e$,

$$F(\vec{u}_p^*, \vec{u}_e^*) \leq F(\vec{u}_p, \vec{u}_e^*) \text{ and } F(\vec{u}_p^*, \vec{u}_e^*) \geq F(\vec{u}_p^*, \vec{u}_e) \tag{7}$$

holds. In other words, it is impossible for the pursuing team and the evading team to improve their performance by changing their control effort.

Next, let us investigate the Hamilton-Jacobi-Bellman-Isaacs framework in detail. Since the order of the min and max operations may be important, two different versions of the dynamic programming principle will be used; namely, the *dynamic programming principle of suboptimality* and the *dynamic programming principle of superoptimality*.

For a more detailed derivation of the dynamic programming principle and the Hamilton-Jacobi-Bellman-Isaacs equation, consult [30].

First, let us define the upper value and lower value in terms of strategies. Given the set of controls *$U_p(t)$* and *$U_e(t)$* a strategy of the pursuing coalition can be described by the mapping *$P: U_e(t) \rightarrow U_p(t)$*, that is, if the evading coalition uses the control *$u_e \epsilon\, U_e(t)$* the strategy *P* will dictate the controls the pursuing coalition should use.

Equivalently, let the evading coalition strategy be described by *$E: U_p(t) \rightarrow U_e(t)$*. Finally, let *$S_p$* and *$S_e$* be the set of all the pursuit and evasion strategies. The modified Bellman's optimality condition can now be described for the differential game framework. For the corresponding expression typically encountered in optimal control theory see Appendix A.

Theorem 1: Bellman's Optimality Condition (Adversarial Case): $\forall(x,t) \in \mathbb{R}^n \times [t_0, T]$ and $\sigma > 0$ such that $(t+\sigma) \in (t_0, T]$,

$$V_{lowervalue}(x,t) = inf_{u_p \in P} sup_{u_e \in U_e} \left\{ \int_t^{t+\sigma} L\big(x(s), s, u_p, u_e\big) ds \right. \\ \left. + V_{lowervalue}\big(x(t+\sigma), t+\sigma\big) \right\} \tag{8}$$

and

$$V_{uppervalue}(x,t) = sup_{u_e \in E} inf_{u_p \in U_p} \left\{ \int_t^{t+\sigma} L\big(x(s), s, u_p, u_e\big) ds \right. \\ \left. + V_{uppervalue}\big(x(t+\sigma), t+\sigma\big) \right\} \tag{9}$$

A proof of theorem 1 can be found in [30]. It should be noted that theorem 1 mirrors Bellman's optimality principle in optimal control which is covered in appendix A.

Next, let us consider how the upper and lower value functions might be found. If the upper value function $V_{uppervalue} \epsilon\, C^1([t_0, T] \times \mathbb{R}^n)$ then $V_{uppervalue}$ satisfies the equality given by:

$$\frac{\partial V_{uppervalue}}{\partial t} + max_{u_e \in U_e} min_{u_p \in U_p} \left\{ \langle \frac{\partial V_{uppervalue}}{\partial \vec{x}}, f\big(\vec{x}, \vec{u}_p, \vec{u}_e, t\big) \rangle \right. \\ \left. + L\big(\vec{x}, \vec{u}_p, \vec{u}_e, t\big) \right\} = 0 \\ V_{uppervalue}(x, T) = g(x) \tag{10}$$

and the lower value function $V_{lowervalue} \in C^1([t_0, T] \times \mathbb{R}^n)$ then $V_{lowerrvalue}$ satisfies the equality ,

$$\frac{\partial V_{lowervalue}}{\partial t} + min_{u_p \in U_p} max_{u_e \in U_e} \left\{ \frac{\partial V_{lowervalue}}{\partial \vec{x}}, f\left(\vec{x}, \vec{u}_p, \vec{u}_e, t\right) \right. \\ \left. + L\left(\vec{x}, \vec{u}_p, \vec{u}_e, t\right) \right\} = 0 \\ V_{lowervalue}(x, T) = g(x) \tag{11}$$

For more detailed discussion of the above result, consult [31].

2.2.1 Minimum Time Differential Games

The work presented in this thesis focuses on problems in which it is the goal of multiple pursuers to intercept multiple evading targets in minimum time. Hence, the cost function considered is given by

$$max_{u_e \in U_e} min_{u_p \in U_p} J = \int_{t_0}^{T} dt \tag{12}$$

The corresponding Hamilton-Jacobi-Bellman-Isaacs equation mirrors that of the minimum-time formulation commonly found in the optimal control literature.

$$min_{u_p \in U_p} max_{u_e \in U_e} \left\{ \langle \frac{\partial V}{\partial \vec{x}}, f\left(\vec{x}, \vec{u}_p, \vec{u}_e, t\right) \rangle \right\} + 1 = 0 \tag{13}$$

The value of the game upon termination is considered zero, that is, $V(x(T),T)$ is zero. However, since time can no longer be used to terminate the optimization problem, a termination constraint must also be specified:

$$\Psi(x(T), T) = 0 \tag{14}$$

The termination constraint is determined based on how the operator defines capture. In the examples provided in the following chapters, a target is considered captured if a pursuer is within a circle of radius r of the target.

Currently, the most commonly used approach to solve the minimum-time problem described by Equations 1, 12, 13, and 14, is to view the problem as a propagating interface problem. Hence, a method known as the Fast Marching Method [32, 33] is an effective technique, derived from the standard Level Set Techniques [34], to solve minimum-time problems quickly.

2.2.2 Current Differential Game Results

Since the 1950s, much work has been done on solving differential games. Since linear problems with quadratic cost functional are relatively simple to manipulate, some interesting results were derived early in the development of the differential game theory framework [35]. In [36] an N-player non-cooperative Linear Quadratic (LQ) differential game was posed as an extended Riccati Differential Equation and solved using standard solution techniques. A comparison study was conducted in [37] in which it was shown that imposing a LQ framework on a problem that is not actually LQ can cause significant loss of performance. A thorough discussion on LQ differential games for which the Riccati equation approach fails along with some illustrative examples can be found in [38].

The LQ game problem with noise introduced into the dynamics has also been studied intensively. In [39], a LQ game with Gaussian noise introduced into the dynamic model was analyzed and it was shown that the players were guaranteed a lower bound on performance given the derived control law. In [40] a thorough introduction to stochastic differential games is provided along with some insights into methods for constructing solutions to stochastic differential games.

The maximum principle was used in [19] and [41] to solve two-player non-linear differential games. The advantage of applying the maximum principle to solving multiplayer differential games lies in the reduced space over which a solution is generated. In [42], the computational burden related to determining efficient pursuit-evasion strategies was reduced by only optimizing over a limited time horizon. The resulting UAV evasion strategy proved to be very promising against a much faster human piloted aircraft.

CHAPTER 3

DECOMPOSITION APPROACH

3.1 Overview of the Approach

As discussed in chapter 2, one very important issue that must be addressed when solving multiplayer differential games using the Hamilton-Jacobi-Bellman-Isaacs framework is how to handle the inherent computational complexity. To address this issue, the decomposition approach eliminates the exponential complexity with respect to the number of players in the game by constructing a collaborative engagement strategy from information derived from solving multiple two-player games. In essence, the multiplayer strategies are constructed by considering a relatively small number of cross sections of the state space and extending the information contained in these cross sections to the rest of the state space.

The decomposition is performed in two stages:

Low-Level:

The interception strategy for each of the pursuers is determined by solving several two-player stochastic differential pursuit-evasion games.

High-Level:

Each pursuer is assigned a target *and* a role. The pursuer’s role can be either to intercept directly or to contain the target. The roles that are assigned to the pursuers depend on a dynamic performance-based region of responsibility (DPRR) derived from the two-player game solution. A representation of the interactions between the stages is shown in Figure 7.

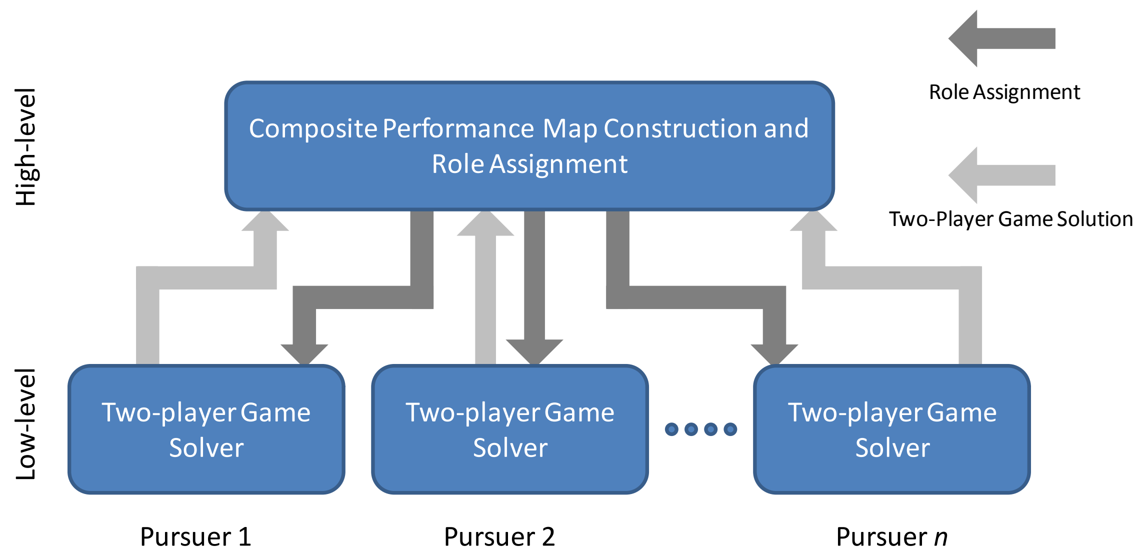
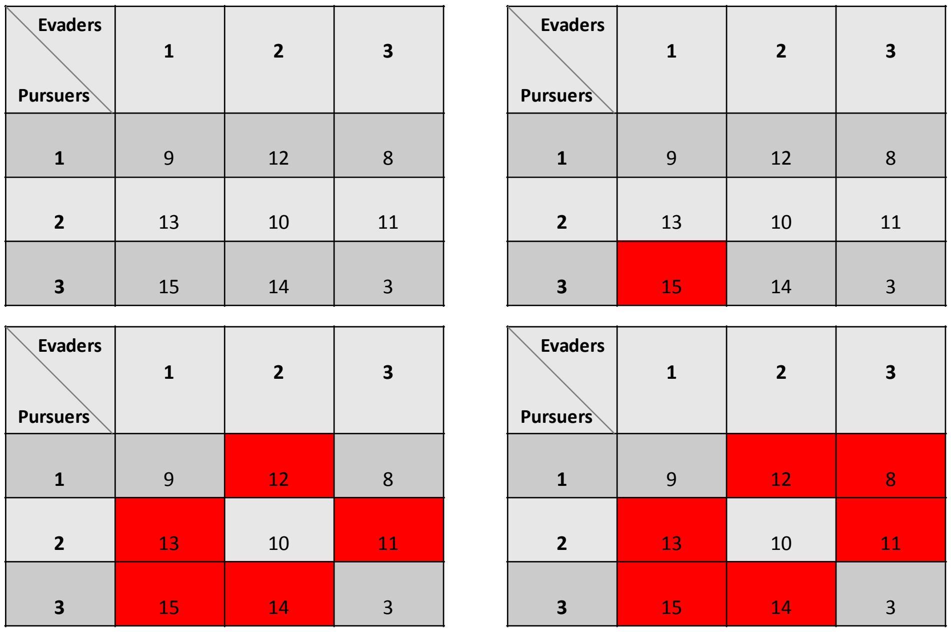


Figure 7. The decomposition of the multiplayer game into several two-player problems.

By performing the decomposition, the cooperation between the pursuing players is not considered. Hence, at the high level of the decomposition approach, cooperation must be reintroduced to improve the performance of the entire team of pursuers. Each of the pursuers is assigned a DPRR based on the estimated interception time derived in the low-level decomposition stage. If an evader is in a pursuer's DPRR, then it is that pursuer's responsibility to intercept the evading target, that is, the pursuer is assigned the "intercept" role. However, if there are no evaders in a particular pursuer's DPRR, then the pursuer will move toward a virtual target in an attempt to contain the target. The virtual target is the point on the boundary of the pursuer's DPRR with the largest difference between the estimated time to capture and the time it takes the evader to reach the point. This point is where it is most likely for the evader to cross into the pursuer's DPRR.

Finally, if there are multiple targets in a single pursuer's DPRR, the assignment strategy becomes more complex. Since the cost of intercepting each target is known for all the pursuers, it is possible to assign "support" roles to the other pursuers. For the minimum time problem considered throughout this work, it is the objective of the "interceptor" and the "support" players to minimize the maximum estimated interception time. Consequently, the worst estimated interception times are removed until a suitable match of pursuers to evading targets is found. A graphical representation of this assignment algorithm is shown in Figure 8.



Evaders / Pursuers	1	2	3
1	9	12	8
2	13	10	11
3	15	14	3

Evaders / Pursuers	1	2	3
1	9	12	8
2	13	10	11
3	🟥 15	14	3

Evaders / Pursuers	1	2	3
1	9	🟥 12	8
2	🟥 13	10	🟥 11
3	🟥 15	🟥 14	3

Evaders / Pursuers	1	2	3
1	9	🟥 12	🟥 8
2	🟥 13	10	🟥 11
3	🟥 15	🟥 14	3

Figure 8. Top Left: The estimated interception times between each pursuer and the corresponding evaders is determined by solving the two-player games. Top Right: The worst case performance estimates are removed. Bottom Left: The removal process is repeated until an assignment must be performed. Bottom Right: The pursuers are assigned to the evaders in the best possible fashion.

Figure 9 illustrates the target assignment. The evading target in the center of the figure is attempting to escape the three pursuers. Only the pursuer at the top of the picture is tasked with intercepting the evading target directly. The other two pursuers are asked to contain the target by heading toward the virtual targets in an attempt to block the evader's possible escape routes. It should be noted that the DPRR for each pursuer is updated regularly; consequently, the tasks assigned to the pursuers also changes frequently depending on the actions taken by the evading target.

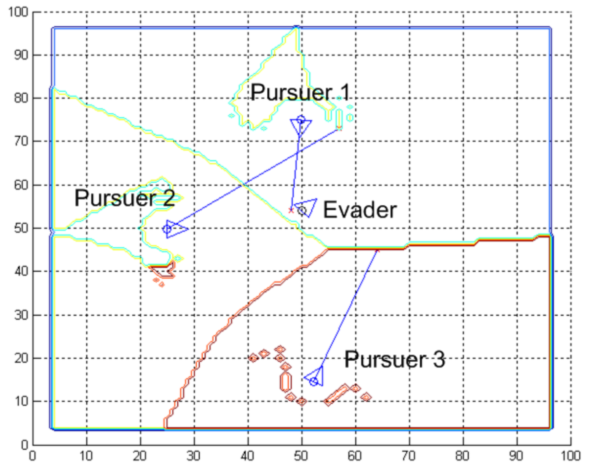


Figure 9. Three pursuers attempt to capture the evading target located at the center. One of the pursuers attempts to intercept the target directly, while the other pursuers perform containment maneuvers.

The decomposition approach is advantageous, since it is computationally much simpler to solve the two-player games than general multiplayer games. The approach is able to adapt rapidly to changes in the scenario. Hence, if new pop-up targets are encountered the roles of each of the pursuers is reassigned to effectively handle the unexpected change to the scenario. Additionally, the algorithm can easily be implemented in a distributed fashion, that is, the computational resources onboard all of the UAVs can be utilized effectively.

3.2 Implementation Considerations

As described in chapter 2, one of the problems encountered when solving differential games is how to find the saddle point in the Hamilton-Jacobi-Bellman-Isaacs equation. Since it is assumed that the solution to the problem is non-trivial, the solution scheme must find the optimal set of controls for the pursuer and the evader numerically. Finding the minimax solution is in itself difficult; however, since the minimax solution has to be found repeatedly, the solution must also be found very fast. To find the solution fast, a gradient descent technique is run four times with randomly selected starting points, and the best overall set of controls from the four iterations is used as the solution to the minimax problem. The multiple starting points are used to ensure that the algorithm does not get stuck at a local minimum or maximum point.

It should also be noted that the pursuers minimize the Hamiltonian before the evader maximizes it. This will provide the evader with a slight advantage, since the pursuers control choice is considered known to the evader, and consequently, the derived solution and the performance estimates are conservative.

One of the advantages of using the decomposition approach to solve stochastic multiplayer differential games lies in the inherent ease with which it can be implemented in a distributed fashion. The distributed implementation approach is shown in Figure 10. The computer onboard the UAVs solves the required two-player differential game, while a central planner combines the information provided by the individual UAVs to determine the best resource assignment. The planner should be located on the UAV with the best onboard computational resources.

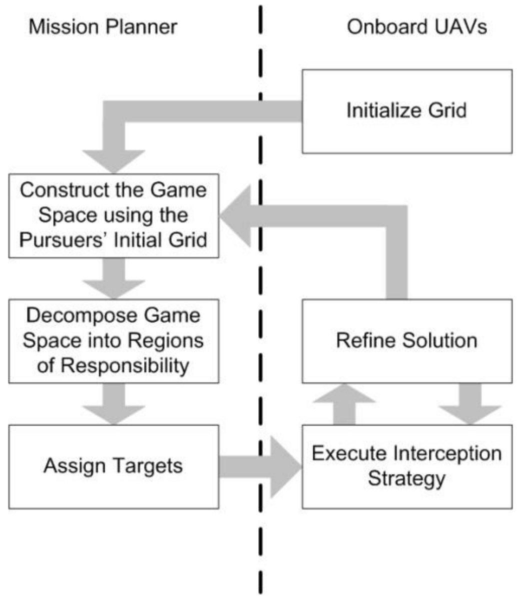

Mission Planner
Onboard UAVs
Initialize Grid
Construct the Game Space using the Pursuers' Initial Grid
Decompose Game Space into Regions of Responsibility
Refine Solution
Assign Targets
Execute Interception Strategy

Figure 10. A Information flow diagram between the UAVs' onboard computers and the mission planner.

The UAVs then execute their assignments as received from the planner and relay refined performance information back to the planner as the interception strategy is executed. As a consequence of this feedback loop, target handoff is implicitly incorporated into the control architecture. It should be noted that one of the bottlenecks in implementing the algorithm in a distributed fashion lies in transmitting the two-player value function between the UAVs and effectively combining the solutions to construct the performance-based regions of responsibility. Therefore, to improve the decomposition approach on current UAV platforms, the derived value function information may have to be compressed to effectively share the performance data between the UAVs.

3.3 *Simulation Results*

The approach was tested using the following system dynamics:

$$\begin{gathered}\dot{x}_{pi} = C_{pi} \cdot \cos(\theta_{pi}) \\ \dot{y}_{pi} = C_{pi} \cdot \sin(\theta_{pi}) \\ \dot{\theta}_{pi} = \omega_{pi} \\ \dot{x}_{ei} = C_{ei} \cdot \cos(\theta_{ei}) \\ \dot{y}_{ei} = C_{ei} \cdot \sin(\theta_{ei}) \\ \dot{\theta}_{ei} = \omega_{ei}\end{gathered} \tag{15}$$

where $(x_{pi}, y_{pi}, \theta_{pi})$ and (x_p, y_p, θ_p) are the position of and heading of the pursuers and the evader respectively. $C_{pi}, C_{ei}, \omega_{pi}, \omega_{ei}$ are the control variables.

For simulation purposes, the system was rewritten in the pursuers' reference frames:

$$\begin{gathered}\dot{x} = -C_P + C_e \cdot \sin(\theta) + \omega_p \cdot y \\ \dot{y} = C_e \cdot \cos(\theta) - \omega_p \cdot x \\ \dot{\theta} = \omega_e - \omega_p\end{gathered}$$

The value function was constructed using a narrow band level set method described in [33]. The motion of each point on the terminal manifold is projected onto a line perpendicular to the terminal manifold. The length of the new vector at point (x, y, θ) is labeled $F_{x,y,\theta}$. $F_{x,y,\theta}$ is essentially the perpendicular propagation speed of the terminal manifold. Then, in an attempt to determine the value of the points in the neighborhood of the manifold, the following relation was used:

$$\begin{bmatrix} \max\left(D^{-x}_{x,y,\theta}V - D^{+x}_{x,y,\theta}V, 0\right)^2 + \max\left(D^{-y}_{x,y,\theta}V - D^{+y}_{x,y,\theta}V, 0\right)^2 \\ + \max\left(D^{-\theta}_{x,y,\theta}V - D^{+\theta}_{x,y,\theta}V, 0\right)^2 \end{bmatrix}^{1/2} = \frac{1}{F_{x,y,\theta}} \quad (16)$$

where

$$D^{-x}_{x,y,\theta}V = \frac{V_{x,y,\theta} - V_{x-1,y,\theta}}{\Delta x} \text{ and } D^{+x}_{x,y,\theta}V = \frac{V_{x+1,y,\theta} - V_{x,y,\theta}}{\Delta x} \quad (17)$$

Since $V_{x,y,\theta}$ is the only unknown in Equation 16, it is computationally simple to determine the values of each of the points in the neighborhood of the terminal manifold. Once the values of the neighboring points have been determined, they are added to the set of points in the terminal manifold. Then, a new neighborhood is constructed around the new manifold, and the process is repeated until the entire value function is constructed. For this example, the game was terminated once the evader was within a circle of radius 10 around the pursuers.

The algorithm can be broken down into the following steps:

Step 1: Label points within the termination condition as “Alive”, and label all the points in the four-connected neighborhood as “Neighbors” and estimate their value. All the other points are labeled as “Far Away”.

Step 2: Select the point amongst the “Neighbors” with the lowest value and label it “Alive”.

Step 3: Add all “Far Away” points in the four-connected neighborhood as “Neighbors” and estimate the value of all the neighbors.

Step 4: Go to Step 2 for as long as there are points labeled “Neighbors”.

The complexity of the algorithm is $O(n \cdot \log(n))$ where n is the number of sample points in the state space.

Figure 11 depicts the output of a single iteration of the decomposition algorithm. The individual value functions for each of the three pursuers are generated, and based on the combined cost map, the pursuers are assigned a region of responsibility. The bottom left graph in Figure 11 shows a very interesting issue that might occur when the value functions are generated. Since the game essentially is bounded by the region over which the value function is constructed, the edge might introduce unforeseen errors. In this case, the edge of the region acts as an obstacle, and it is very difficult for the pursuer to capture the target if it is located directly behind it.

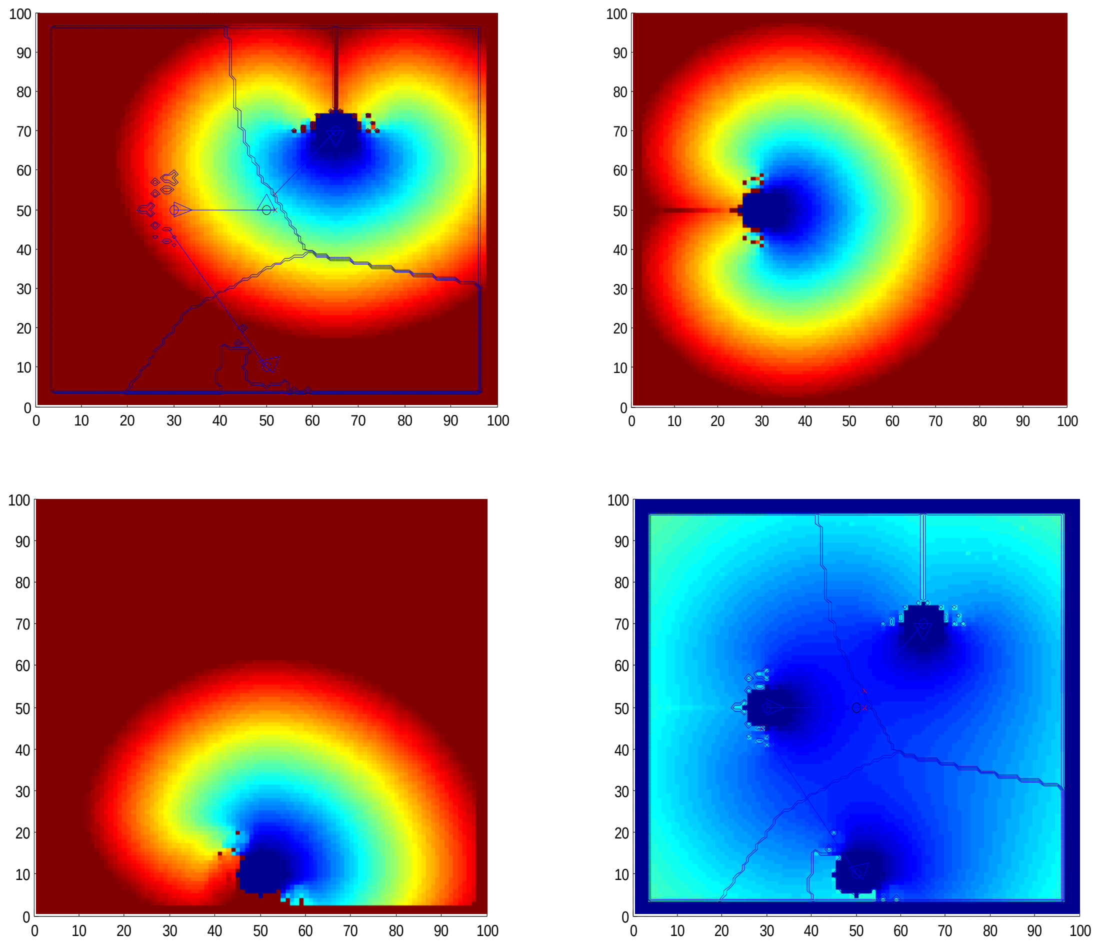


Figure 11. Top Left: The regions of responsibility are superimposed on a plot of the pursuer 1's value function. Low value regions are blue, while high value regions are red. Top Right and Bottom Left: Pursuer 2 and 3's value functions. Bottom Right: The combined value function map with the regions of responsibility superimposed.

In Figure 12, the dynamic nature of the combined value function map is shown. Notice, that the DPRRs are not necessarily connected, especially if the vehicle dynamics are non-linear, so determining best point to intercept the target must be implemented very carefully.

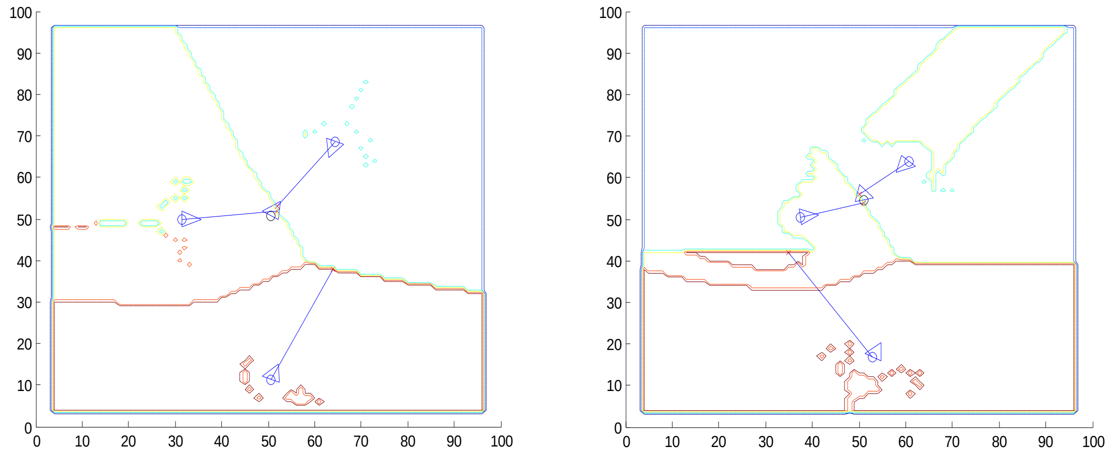


Figure 12.The left and the right graphs depict the DPRRs at 3 seconds and 15 seconds after the interception is commenced.

Figure 13 depicts the difference between the estimated interception time and the time the pursuers actually needed to intercept the target. Ideally, the estimated interception curve should be a line as indicated by the yellow curve. However, due to the conservative estimate of performance and discretization errors, the estimate is much higher than the ideal interception time.

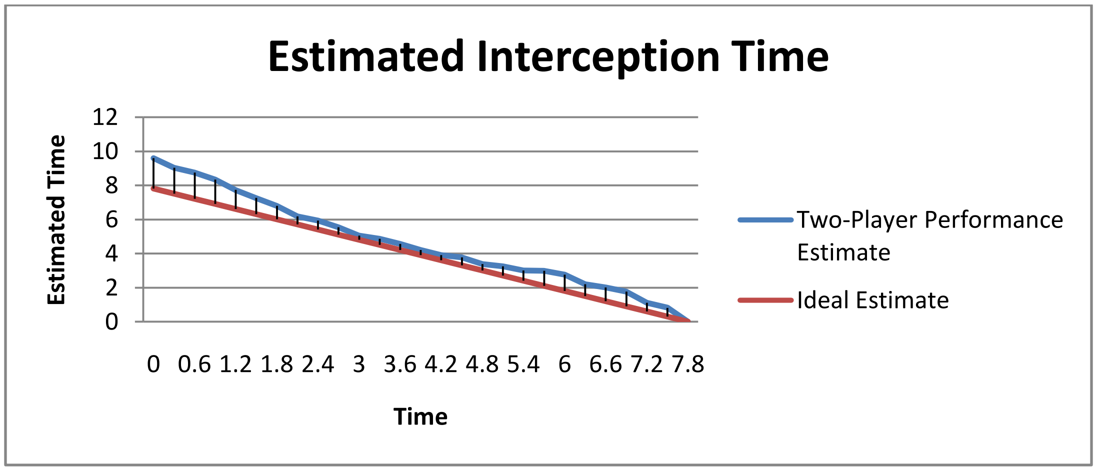


Figure 13. Performance Graph: The interception estimate is higher than the ideal curve due to the conservative nature of the approach.

3.4 Important Additional Considerations

When the differential game was decomposed into several two-player differential games, one implicit but very important assumption was made; specifically, it was assumed that the two-player differential games had a solution. This detail becomes very important in situations where the evading targets are significantly faster than the pursuers. In such situations the estimated interception time becomes infinite, and hence it is impossible for the planner to determine the DPRRs. Consequently, all the pursuers will attempt to intercept the evader directly which may not be the best interception strategy.

3.5 Performance Considerations

To illustrate a couple of interesting aspects that arise when analyzing multiplayer differential games, let us consider the three player game consisting of two pursuers and a single evading target played on the line. This example is very simple; however it also illustrates a number of interesting problems that must be considered when analyzing multiplayer games. Assuming that the evading target is captured if it is within one unit of either of the two pursuers, then the game termination points is described in three dimensions by the solid depicted in Figure 14.

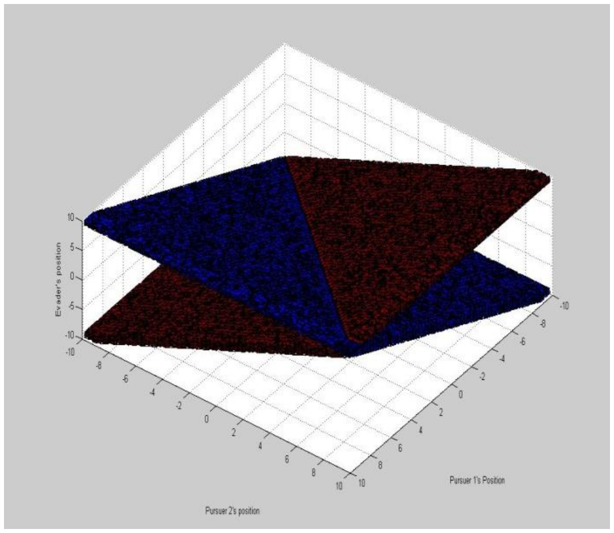


Figure 14. The termination states of the three player game is described by a set of plates with a thickness of two units.

The blue "plate" describes the points in the state space for which Pursuer 1 terminates the game and the red describes the points for which Player 2 terminates the game. The intersection of the two plates describes the set of points for which the two pursuers collaboratively captures the target. To simplify the discussion a little, the next couple of illustrations will describe the state of the game in the evaders reference frame. By rewriting the system in this fashion, it is possible to illustrate the game in the plane as shown in Figure 15. The system considered in this simple example is described by

$$\begin{gathered} \dot{x}_{Pi} = u_{Pi}, i = 1{,}2 \\ \dot{x}_E = u_E \end{gathered} \tag{18}$$

where u_{Pi} = [-1, 1] and u_E = [-.9, .9].

Notice that only four points on the termination surfaces correspond to situations where multiple players terminate the game in a collaborative fashion. The situations where the game is terminated on the two points that are located in the first and third quadrant is trivial, however the two points that are located in the second and fourth quadrant provide significant insight into multiplayer differential games.

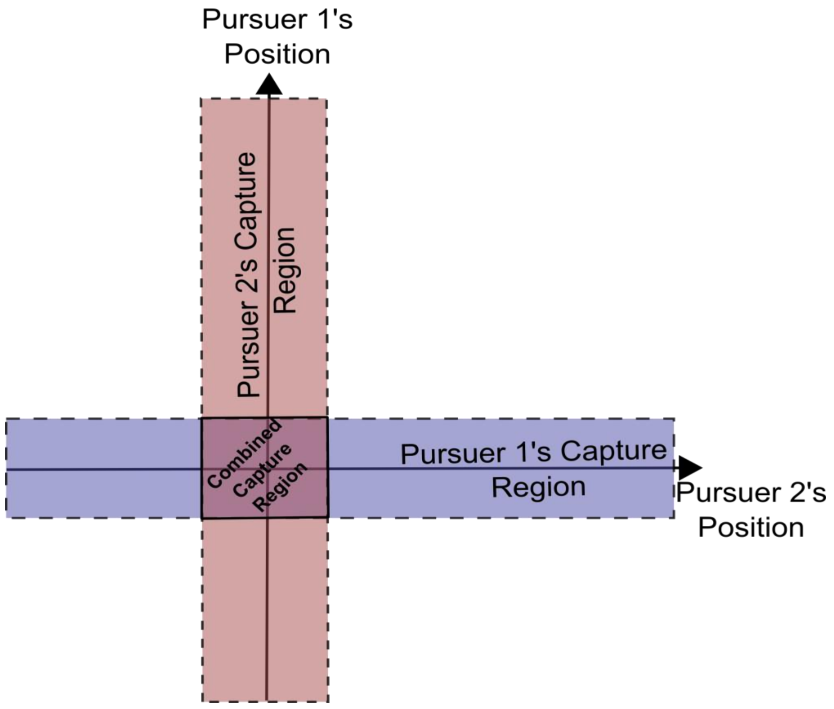


Figure 15. Capture regions described in the evader's reference frame.

Due to the symmetry of this problem, only the case where the game is terminated on the termination point in the second quadrant will be analyzed (the arguments are exactly the same for the fourth quadrant case). The region between the termination regions is divided into smaller sections as shown in Figure 16. The green “Pure Two-Player Solution” regions are the states within which the evading target can completely ignore one of the two pursuers, that is, the evading target starts out too close to one of the pursuers, and will therefore not reach the other player no matter what strategy the evader uses. Pursuer 1’s and Pursuer 2’s advantage regions describe the set of states within which the evading target is closer to being captured by Pursuer 1 or Pursuer 2 respectively; however, the evading target cannot disregard the other player since both players might be able to terminate the game depending on the strategy used by the evading target. Finally, the states along the balanced three-player game solution, describe situations where the evading target is equally close to both the two pursuers.

One interesting aspect of the multiplayer game played on the line can be observed in the control effort of the evading target. If the game is in either the advantage regions or in the balanced three player solution region, it does not matter what control decision the evading target makes as long as the target remains in the multiplayer region of the game’s state space. This aspect of the game is illustrated in Figure 17. The evading target can either move to the center point between the two pursuers directly as indicated by strategy 2, or the player can move around a little before returning to the center as indicated by strategy 1. In essence, there are multiple optimal trajectories that the evading target can follow; however, there is only a single optimal strategy for the pursuers.

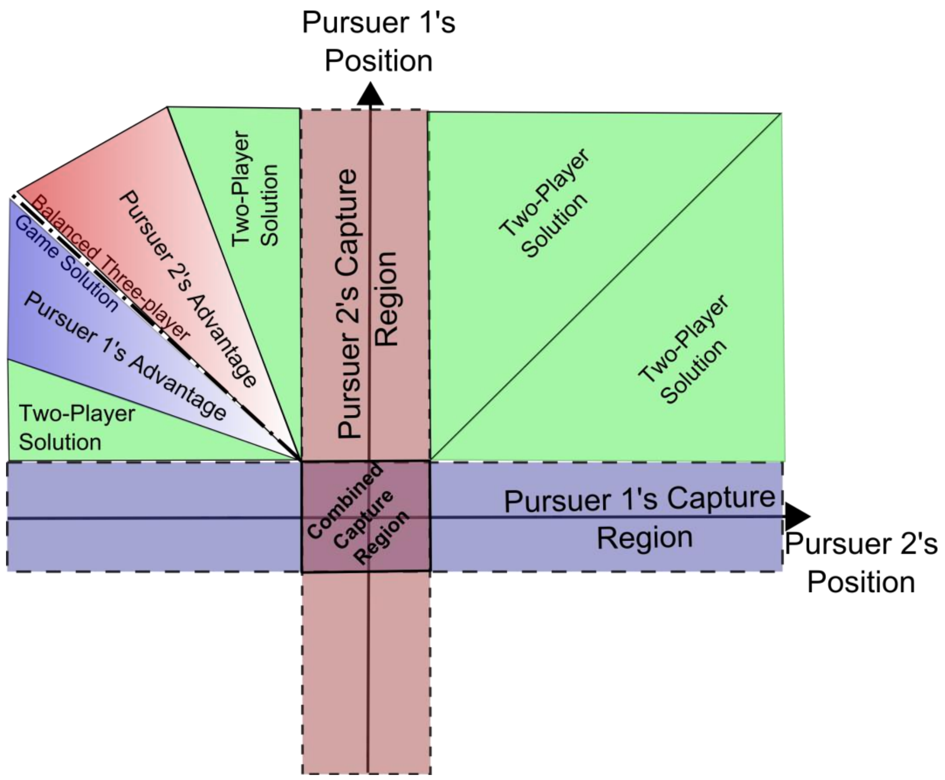


Figure 16. The state space between the capture regions is divided into multiple regions interesting regions.

To see that the target's evasion strategy is not unique, consider the Hamilton-Jacobi-Bellman-Isaacs equation:

$$\begin{gathered} max_{u_e}\Big\{min_{u_{p1}}\langle\frac{\partial V_{multiplayer}}{\partial x_{p1}}, f_{p1}(x_{p1},u_{p1}) - f_e(x_{p1},u_e)\rangle + \\ min_{u_{p2}}\langle\frac{\partial V_{multiplayer}}{\partial x_{p2}}, f_{p2}(x_{p2},u_{p2}) - f_e(x_{p2},u_e)\rangle\Big\} + 1 = 0 \end{gathered} \tag{19}$$

Equation 19 only holds if the game is terminated by both of the pursuing players, that is, the boundary condition for the game played in the second quadrant is the point (-1,1). Since the boundary condition is non-differentiable, determining the spatial derivatives of the value function on the boundary becomes difficult. In this case the partial derivatives can have any value in the ranges $0 < \frac{\partial \mathrm{V_{multiplayer}}}{\partial \mathrm{x_{p1}}} < \frac{\partial \mathrm{V_{two-player}}}{\partial \mathrm{x_{p1}}}$ and $\frac{\partial \mathrm{V_{two-player}}}{\partial \mathrm{x_{p2}}} < \frac{\partial \mathrm{V_{multiplayer}}}{\partial \mathrm{x_{p2}}} < 0$. For instance, strategy 1 in Figure 17 corresponds to the termination condition where $\left|\frac{\partial V_{\text{multiplayer}}}{\partial x_{p2}}\right| > \left|\frac{\partial V_{\text{multiplayer}}}{\partial x_{p1}}\right|$, while strategy 2 corresponds to the solution where $\left|\frac{\partial V_{\text{multiplayer}}}{\partial x_{p2}}\right| < \left|\frac{\partial V_{\text{multiplayer}}}{\partial x_{p1}}\right|$.

In cases where a continuous and differential multiplayer termination surface is provided, it becomes much simpler to determine the best possible termination of the game. Such a scenario is shown in Figure 18. Not only do the pursuers terminate the game at different states, but the spatial derivative of the value function around this point is also unique.

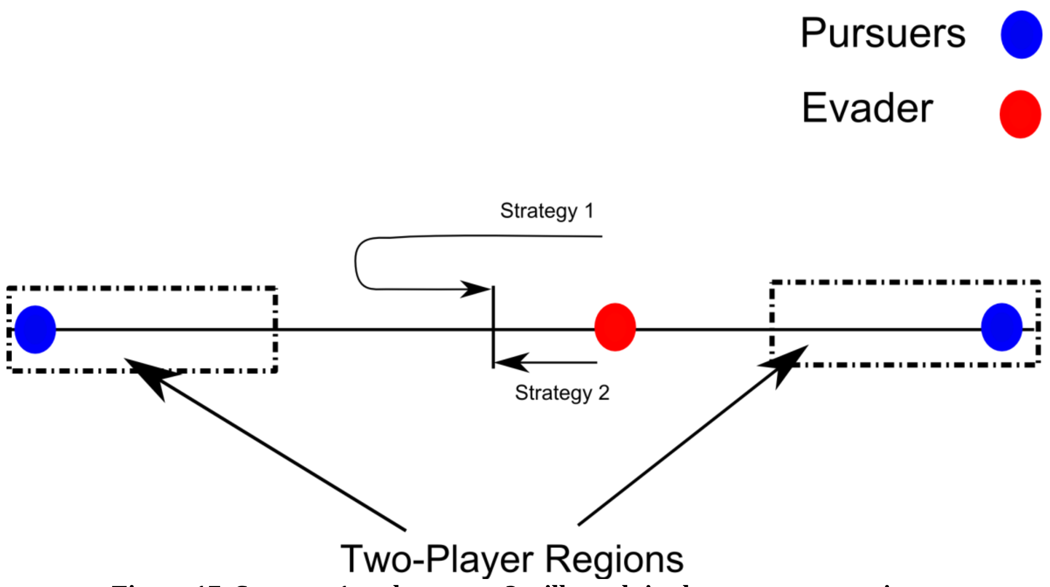


Figure 17. Strategy 1 and strategy 2 will result in the same capture time.

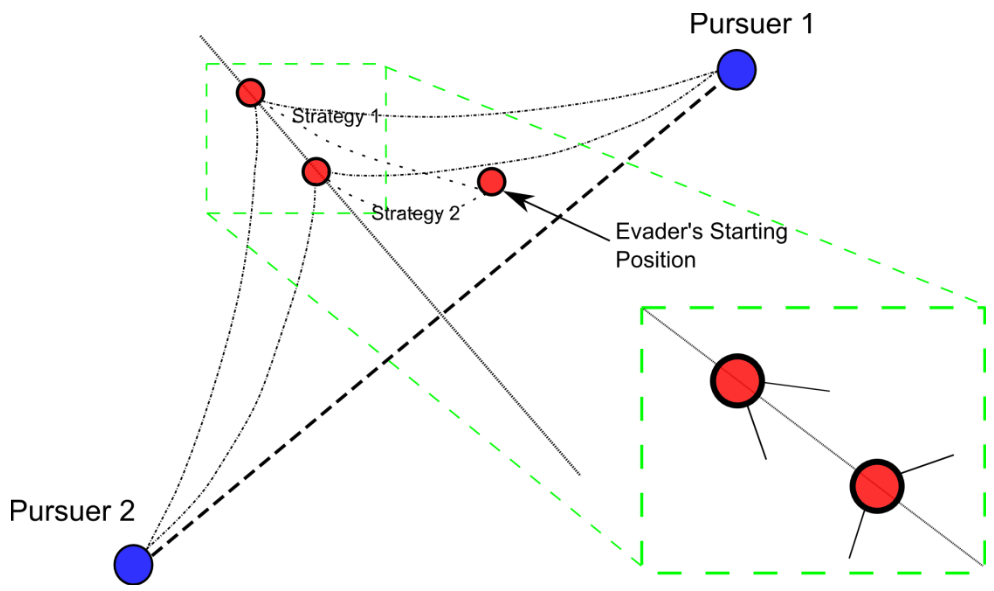


Figure 18. The evader's strategy influences the pursuers' approach angle at termination.

3.5.1 Performance Estimates

The multiplayer value function will be different from the combined value function; however, as shown in Figure 16, for large regions of the multiplayer game state space the two value functions are equivalent. Determining the performance of cooperative multivehicle systems is a non-trivial exercise, and due to the assignment algorithm utilized when multiple targets are encountered, the overall performance of the decomposition algorithm cannot be guaranteed. Additional details on the assignment algorithm and particularly why it is inherently suboptimal is provided in chapter 4. However, it is possible to establish some performance guarantees when multiple pursuers are attempting to intercept a single target. Again, it is assumed that the solution to the two-player differential game exists, but in addition to this assumption, it is also assumed that the evading target performs optimally, the value functions are continuous, and the pursuers' dynamics are uncoupled.

Since the decomposition algorithm ensures that the target will be captured, it may be of interest to determine how accurate the estimate of the interception time is. Let us consider the case in which *n* pursuers are attempting to intercept a single evading target.

In this case, the system of two-player games for the *n* pursuers along with the individual termination condition can be written as:

$$\begin{gathered}
min_{u_{p_1}\in U_p} max_{u_e\in U_e}\left\{\langle\frac{\partial V_{p1}}{\partial \vec{x}_{p1}}, f\left(\vec{x}_{p_1},\vec{u}_{p_1},\vec{u}_e,t\right)\rangle\right\}+1=0,\\
V_{p1}\left(\vec{x}_{p_1}(T),T\right)=0,\\
min_{u_{p_2}\in U_p} max_{u_e\in U_e}\left\{\langle\frac{\partial V_{p2}}{\partial \vec{x}_{p2}}, f\left(\vec{x}_{p_2},\vec{u}_{p_2},\vec{u}_e,t\right)\rangle\right\}+1=0,\\
V_{p2}\left(\vec{x}_{p_2}(T),T\right)=0,\\
\vdots\\
min_{u_{p_n}\in U_p} max_{u_e\in U_e}\left\{\langle\frac{\partial V_{pn}}{\partial \vec{x}_{pn}}, f\left(\vec{x}_{p_n},\vec{u}_{p_n},\vec{u}_e,t\right)\rangle\right\}+1=0,\\
V_{pn}\left(\vec{x}_{p_n}(T),T\right)=0,
\end{gathered}\tag{20}$$

and the complete game formulation for the multiplayer game is:

$$\begin{gathered}
min_{u_p\in U_p} max_{u_e\in U_e}\langle\frac{\partial V_T}{\partial \vec{x}}, f\left(\vec{x},\vec{u}_p,\vec{u}_e,t\right)\rangle+1=0,\\
V_T(\vec{x}(T),T)=0
\end{gathered}\tag{21}$$

where u_p is the control of all the pursuers, u_{pi} for i = 1, 2, …, n is the part of the control vector u_p that player *i* has access to, V_T is the total value function, V_{pi} is the value function for the two-player game consisting of the evading target and pursuer *i*, $\vec{x}$ is the complete state space, and $\vec{x}_{pi}$ is the state of pursuer *i* with respect to the evading target.

The value function generated by the decomposition algorithm is pieced together by the two-player value functions, that is,

$$V_{decomp}=min\left[V_{p1},V_{p2},\dots,V_{pn}\right]<\infty \tag{22}$$

An important question is: What is the discrepancy between the approximated value function V_{decomp} and the total value function V_T?

First, to simplify the notation, let us assume that u_e is always chosen optimally; hence, the *max* term in Equation 20 and 21 will be implied. Due to the independence of the $f_{pi}()$'s it is possible to rewrite Equation 21 as:

$$\begin{gathered}
min_{u_{p_1}\in U_p}\langle\frac{\partial V_{p1}}{\partial \vec{x}_{p1}}, f\left(\vec{x}_{p_1},\vec{u}_{p_1},t\right)\rangle+min_{u_{p_2}\in U_p}\langle\frac{\partial V_{p2}}{\partial \vec{x}_{p2}}, f\left(\vec{x}_{p_2},\vec{u}_{p_2},t\right)\rangle\\
+\cdots+min_{u_{p_n}\in U_p}\langle\frac{\partial V_{pn}}{\partial \vec{x}_{pn}}, f\left(\vec{x}_{p_n},\vec{u}_{p_n},t\right)\rangle+1=0,\\
V_T\left(\vec{x}_{p_1}(T),\vec{x}_{p_2}(T),\dots,\vec{x}_{p_n}(T),T\right)=0
\end{gathered}\tag{23}$$

Consider the case where pursuer *i* takes part in the termination of the game, then we know that $V_{pi}\left(\vec{x}_{p_i}(T), T\right) = V_T\left(\vec{x}_{p_1}(T), \vec{x}_{p_2}(T), \ldots, \vec{x}_{p_n}(T), T\right)$.

Lemma 1: Given that pursuer *i* terminates the game, $\exists\, \vec{x}_{pi} - \delta$, where $0 < \delta \ll 1$ such that for $V_{pi}\left(\vec{x}_{pi} - \delta, t\right), \exists\, \varepsilon \geq 0$ which satisfies,

$$V_{pi}\left(\vec{x}_{pi} - \delta, t\right) = V_T\left(\vec{x}_{p_1} - \delta, \vec{x}_{p_2} - \delta, \ldots, \vec{x}_{p_n} - \delta, t\right) + \varepsilon \tag{24}$$

Proof of lemma 1: This proof is very simple. If Lemma 1 is not true, then at point $\vec{x}_{pi} - \delta$ there exist an ε < 0 such that for at least one pursuer (say pursuer *i*), $\langle \frac{\partial V_T}{\partial \vec{x}_{pi}}, f\left(\vec{x}_{p_i}\vec{x}_{pi} - \delta, \vec{u}_{p_i}, t\right)\rangle > 0$. However, since a pursuer can always be removed from the game, pursuer *i*'s best strategy is not to play at all. Hence, the contribution of all the pursuers will at most be 0, that is, $\langle \frac{\partial V_T}{\partial \vec{x}_{pi}}, f\left(\vec{x}_{p_i}, \vec{u}_{p_i}, t\right)\rangle \leq 0$, so $\varepsilon \geq 0$.

As a consequence of lemma 1, and Bellman's optimality principle, we see that $V_T \leq V_{pi} \forall i = 1,2,\ldots,n$ which implies that $V_T \leq V_{decomp}$.

Next, let us consider how the error accumulates over time. At point $\vec{x}_{pi} - \delta$ we know that *ε = 0* if pursuer *i* terminates the game alone, since $\langle \frac{\partial V_T}{\partial \vec{x}_{pj}}, f\left(\vec{x}_{p_j}, \vec{u}_{p_j}, t\right)\rangle = 0 \forall j \neq i$. The bound on ε is $\varepsilon \approx (\bar{f} - \tilde{f}) \cdot \delta$ where $\tilde{f}$ and $\bar{f}$ are the change in states with respect to time for pursuer *i* for the multiplayer and the two-player game respectively, where pursuer *i* is the pursuer that dictates the value V_{decomp} for a given state. This bound is determined by considering what happens if all the pursuers intercept the target simultaneously. Then the *n* inner products in Equation 23 are all non-zero. However, since all the pursuers intercept the target simultaneously, the contributions from each of the inner products is equal, that is,

$$\langle \frac{\partial V_T}{\partial \vec{x}_{pi}}, f\left(\vec{x}_{p_i}, \vec{u}_{p_i}, t\right)\rangle = \langle \frac{\partial V_T}{\partial \vec{x}_{pj}}, f\left(\vec{x}_{p_j}, \vec{u}_{p_j}, t\right)\rangle, \forall i,j \in \{1,2,\ldots,n\} \tag{25}$$

Since the contribution to the total value from each of the pursuers is equal, the control effort changes, that is, $\bar{f} \neq \tilde{f}$. Consequently, for pursuer *i* to reach a particular state in the delta neighborhood of the termination manifold, the difference in the two value functions become:

$$V_{pi}\left(\vec{x}_{p_i}(T-\delta), T-\delta\right) - V_T\left(\vec{x}_{p_1}(T-\delta), \vec{x}_{p_2}(T-\delta), \dots, \vec{x}_{p_n}(T-\delta), T-\delta\right) \leq (\bar{f} - \tilde{f}) \cdot \delta \qquad (26)$$

It is difficult to extend this error bound in the neighborhood of the terminal manifold to the entire state space since the two trajectories generated by the multiplayer and two-player game algorithms are different. Hence, it is not possible to integrate the error along the trajectory and compare the accumulated difference.

3.6 Problems with Stochastic Disturbances

Consider a scenario in which the dynamics of the pursuing and the evading vehicles are subject to some disturbances either from internal measurement error or from external sources. If the noise is assumed to be accurately characterized by a standard Brownian motion *W(t)* (with *W(0) = 0*), the dynamics may be described by,

$$dx = f\left(t, x, u_p, u_e\right)dt + \sigma(t, x, u_p, u_e)dW(t).$$

where *x* is the state of the game, u_p and u_e are the controls of the pursuers and the evaders respectively and *σ(·)* is the diffusion functional. The corresponding minimum-time generalized Hamiltonian is given by:

$$\begin{aligned}\bar{H} = \underset{u_p}{min}\underset{u_e}{max}\left(\langle\frac{\partial V_T}{\partial \vec{x}}, f(x, u_p, u_e, t)\rangle + 1\right. \\ \left. + \frac{1}{2} tr\left[\sigma(t, x, u_p, u_e)^T \sigma(t, x, u_p, u_e) \cdot \frac{\partial^2 V_T}{\partial \vec{x}_i \partial \vec{x}_j}\right]\right)\end{aligned} \qquad (27)$$

where *i, j = 1,2, …n*.

The second order term given in Equation 27 will in many cases help alleviate some of the numerical difficulties often encountered when solving the Hamilton-Jacobi-Bellman-Isaacs equation. The second order term of cause imposes additional smoothness constraints on the boundary conditions; however, due to the diffusive nature of the second order term, potential discontinuities may be smoothed.

To determine the numerical solution to the problem described by Equation 27, it is necessary to modify the fast marching scheme slightly. One of the main difficulties lies in ensuring that there are enough support points to determine the solution, that is, in the deterministic case, it was possible to estimate the interception time at a given point based on one point on the propagating surface. With the second-order partial differential equation, it is necessary to have at least two support points as shown in Figure 19. To ensure that the fast marching algorithm will work for the stochastic version of the problem, termination surface is propagated backwards twice. During the first pass, if there are any points for which it is impossible to estimate the second-order derivatives, the value for that point is determined using only the deterministic version of the problem. During the second pass, the second-order derivatives at the same problem points are determined based on the value from the first pass. Even though it is possible to achieve a better approximation to the solution by sweeping multiple times, the sweep is only performed twice in an attempt to reduce the overall computational complexity.

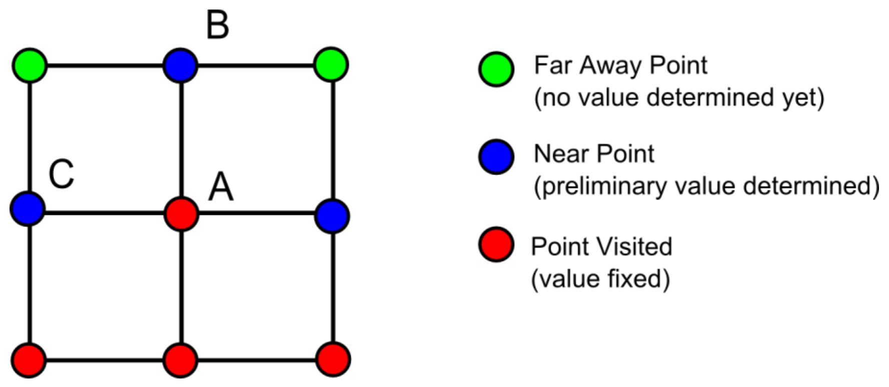


Figure 19. Estimating the value of point B based on point A is possible since there is an additional point behind point A, while estimating the value of point C based on point A is not possible.

3.7 Simulation Example

To illustrate the smoothing properties of the stochastic terms in the differential game formulation, the value functions for a deterministic two-player game and a corresponding stochastic two-player game are constructed. For simplicity the variance was chosen to be constant throughout the game. However, to emphasize the differences in the two value functions, the state space was approximated with a low number of samples. The deterministic value function is shown in Figure 20. The collapsed representation of the value function shown at the top of the figure appears to be relatively smooth; however, when one takes a closer look at the cross-sections of the value function, significant discontinuities appear.

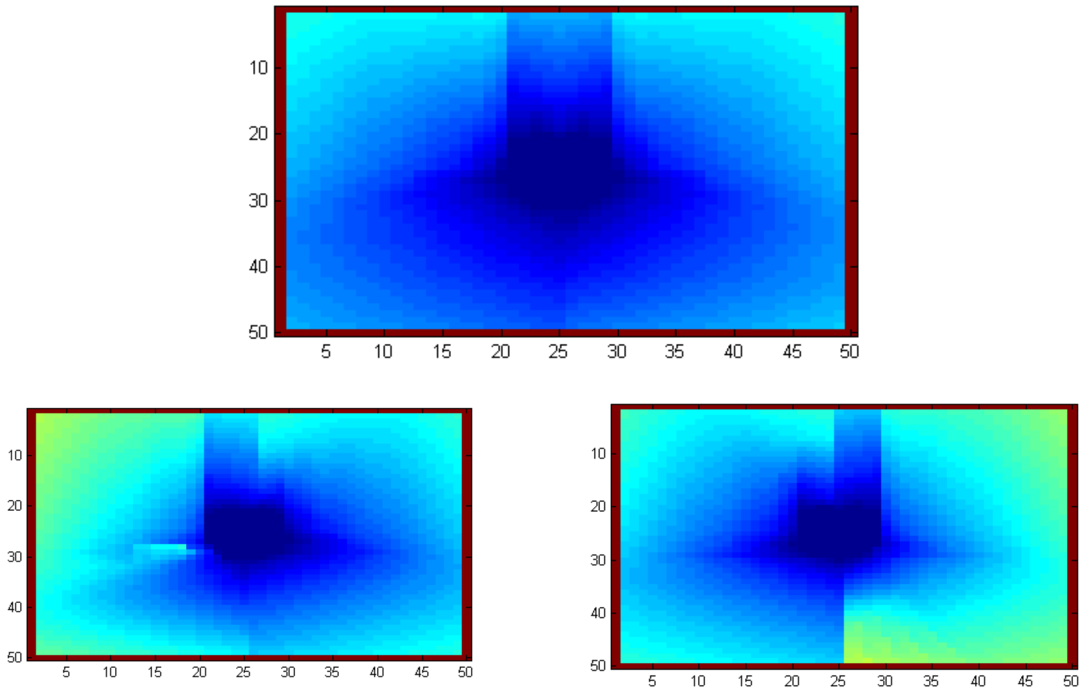


Figure 20. Top: The two-player value function collapsed using the MIN operator. Bottom: Two cross-sections of the four-dimensional value function.

The value function corresponding to the same two-player game with a disturbance term introduced into the system dynamics is depicted in Figure 21. The value function is significantly smoother due to the diffusive effect of the stochastic term introduced into the system dynamics.

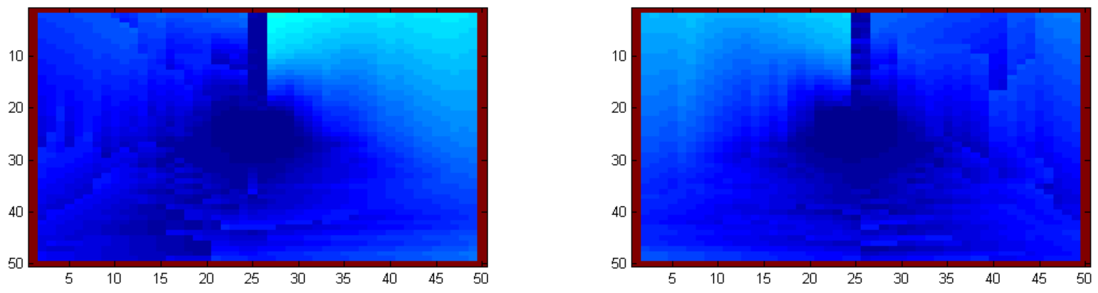


Figure 21. Cross-section of the four-dimensional value function with stochastic disturbances.

It should be noted that there are still numerical problems due to the low spatial resolution chosen, however the diffusive properties of the stochastic term significantly smoothes the discontinuities.

CHAPTER 4

MAXIMUM PRINCIPLE APPROACH

Attempting to solve the Hamilton-Jacobi-Bellman-Isaacs equation for a multi-player differential pursuit-evasion game is very time consuming, mainly due to the space over which the value of the game must be found. In an attempt to combat the complexity of the problem, the approach to solving multiplayer differential games presented in chapter 3, decomposed the problem into several two-player differential games which ensures that certain types of problems can be solved rapidly. However, in situations in which the vehicle dynamics can only be accurately described using a large number of states, even the decomposition approach becomes computationally infeasible for real-time applications.

Instead of relying on dynamic programming techniques to solve the problem, the *maximum principle* approach can be used to determine good interception strategies. The maximum principle generates a characteristic strip of the value function found by solving the Hamilton-Jacobi-Bellman-Isaacs equation. Hence, by using the current state of the game and the termination condition, it is possible to generate a solution to the differential game problem without having to reconstruct the entire value function.

One of the important challenges that must be considered when utilizing the maximum principle to solve a differential game is related to the difficulty encountered when attempting to accurately describe the boundary conditions of the set of differential equations that must be solved. If the maximum principle is applied correctly, the set of n differential equations will have n boundary conditions. Unfortunately, some of the n boundary conditions are given at the beginning of the game, while the rest are given at termination. Hence, to derive a solution to most differential games, it is necessary to use numerical solution techniques such as the shooting method to convert the two-point boundary value problem to either an initial or terminal value problem [43, 44, 45].

Another problem that must addressed when the maximum principle is applied to the multiplayer differential game problem is how to plan the complete interception strategy.

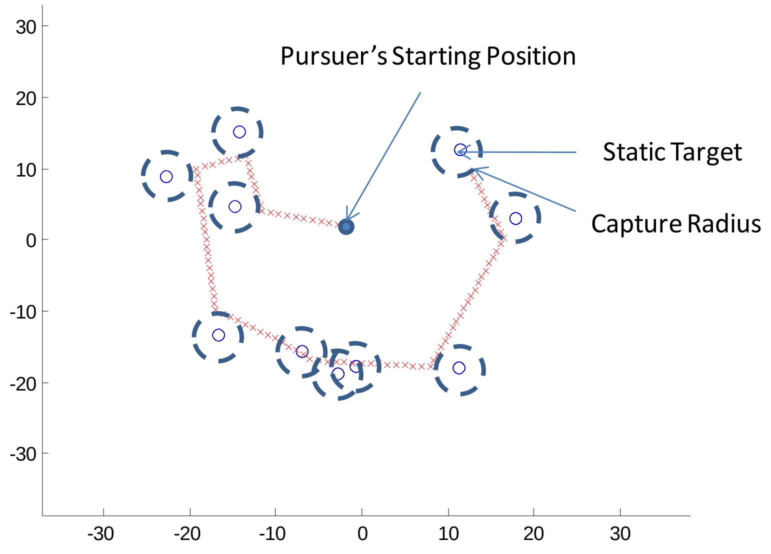


Figure 22. Greedy algorithm applied to an interception problem with 10 static targets and a single intercepting vehicle.

A simple interception plan for a single pursuer and ten static targets is shown in Figure 22. The strategy used by the pursuer is a simple greedy-type algorithm, that is, the pursuer will move toward the closest target until it is captured and then move on to the next. On the other hand, if the pursuer plans the entire trajectory as shown in Figure 22, not only does the pursuer have to consider all the possible ways that the evading targets can be captured, but the evading targets will also be able to utilize the pursuer’s long-term plan to help them improve their collective performance. Consequently, to avoid having the targets gain a significant advantage, the pursuer must replan continuously which becomes computationally infeasible for even the simplest cases.

To ease this computational burden, a multiplayer greedy-type assignment algorithm is used to assign targets to the pursuers. The matching problem is setup as a bipartite graph matching problem as shown in Figure 23. The pursuers are assigned to the target that it can intercept the fastest while ensuring that the entire team of pursuers attempt to intercept as many targets as possible.

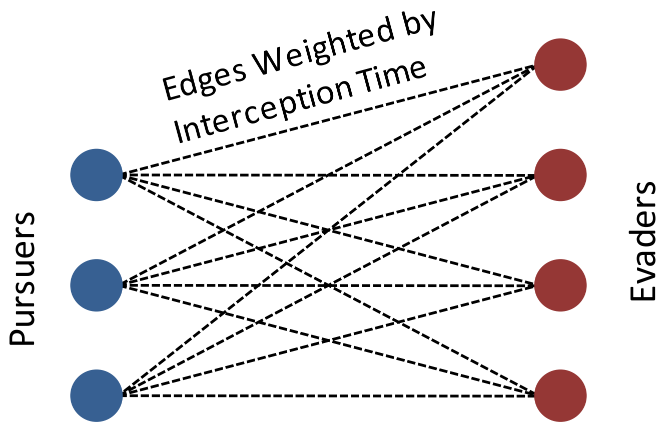


Figure 23. Graph representation of the target assignment problem.

An outline of the assignment algorithm that was used to test the performance of maximum principle approach is shown in Table 1.

Table 1. Assignment algorithm used to assign pursuers to the evading targets.

Step	Description
1	Determine the weights in the bipartite graph
2	Remove the highest weight while remembering the pursuer and the evader associated with the weight.
-	If one of the evader vertices become isolated then:
3A	If the number of evaders is less than or equal to the number of pursuers, then assign the pursuer associated with the removed weight to the evader and remove the pursuer vertex.
3B	Else remove the evading target vertex from the problem.
5	If a pursuer vertex becomes isolated then assign he pursuer to the evader.
6	Go to step 2 until all the pursuers have been assigned a target.

Since a global interception plan is not constructed, the evading targets are provided with four evasion strategies:

1. The evaders attempt to increase the distance to the closest pursuer.
2. The evaders attempt to increase the distance to the closest evader.
3. The evaders attempt to increase the distance to the closest targeted evader. If an evader is targeted, then it will increase the distance to the closest pursuer.
4. The evaders move away from the center of the swarm of evaders.

The term *distance* in the above list of strategies do not refer to Euclidian distances, but rather the time it takes for a pursuer to cover a given gap either between a pursuer and an evader or between two evaders.

The first strategy appears to be rather ineffective when considering multiple evading targets, due to a lack of dispersal of the targets. Strategies two and four have very nice dispersal properties, but the evading targets tend to get captured quickly since many of them will run directly into the pursuers.

The best strategy appears to be strategy three, since it reflects an integral property of the interception problem. In most multiplayer problems, a pursuer will capture one of the evading targets first. Hence, by increasing the distance between the evading targets while not moving directly into the pursuers' interception trajectories, it is possible for the targets to increase the capture.

A simple example of the four interception strategies is shown in Figure 24. The pursuers and the evaders are able to change their velocity vector instantaneously; consequently, the Euclidian distance is proportional to the interception time. The example consists of two pursuing and twenty evading players. The pursuers are 33% faster than the evading targets.

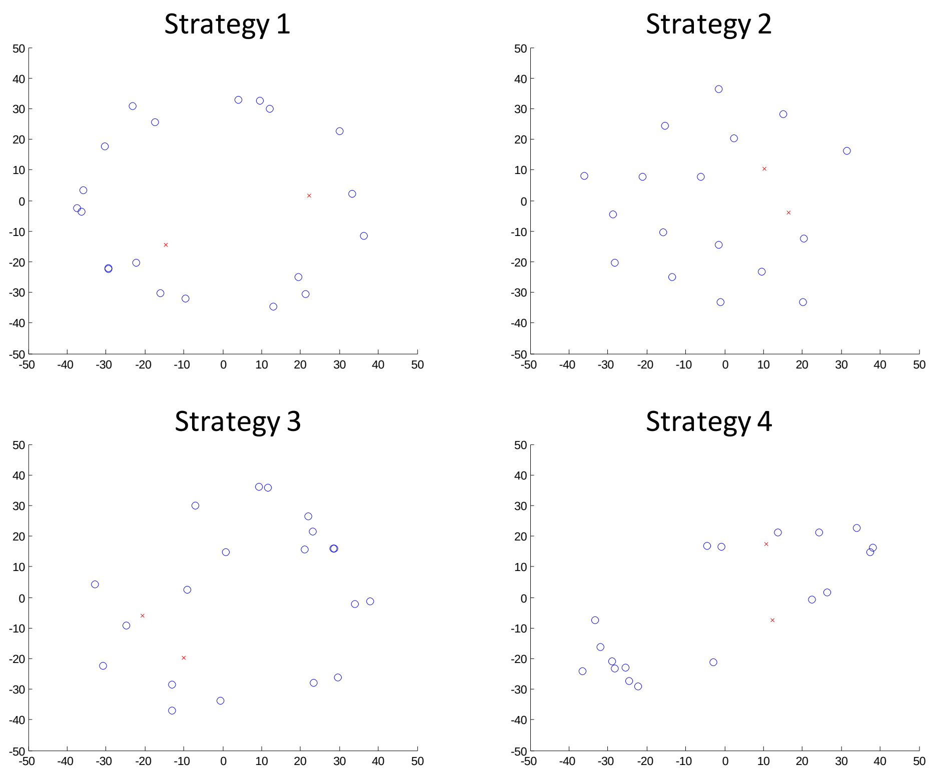


Figure 24. The four plots illustrate the distribution of the evading targets (indicated by the blue circles) after five time units of simulation using the four different evasion strategies. In each of the four examples the two pursuing players start at a random position within a circle of radius of two units and centered on the origin. The 20 evading targets were randomly in band centered on the origin and the distance to the origin between 15 and 25 units.

The graphs shown in Figure 24 provide some insight into how the different evasion strategies might perform given the prescribed pursuit strategy. However, to gain some additional insight into these different evasion strategies it is useful to compare actual interception times. The first set of trials consists of five pursuing players intercepting a variable number of evading targets. Since the initial deployments are randomly chosen, each strategy simulation was executed a thousand times. The first performance graph shown in Figure 25 shows the interception times with the pursuers deployed within a circle centered on the origin. The evading targets were deployed at a distance between 13 and 23 units away from the pursuers' deployment circle. The second performance graph shown in Figure 26 shows the interception times the same scenario, except the pursuers and evading targets are all deployed within the same circle with a radius of 25 and centered on the origin. A depiction of the initial deployment regions is provided in Figure 27.

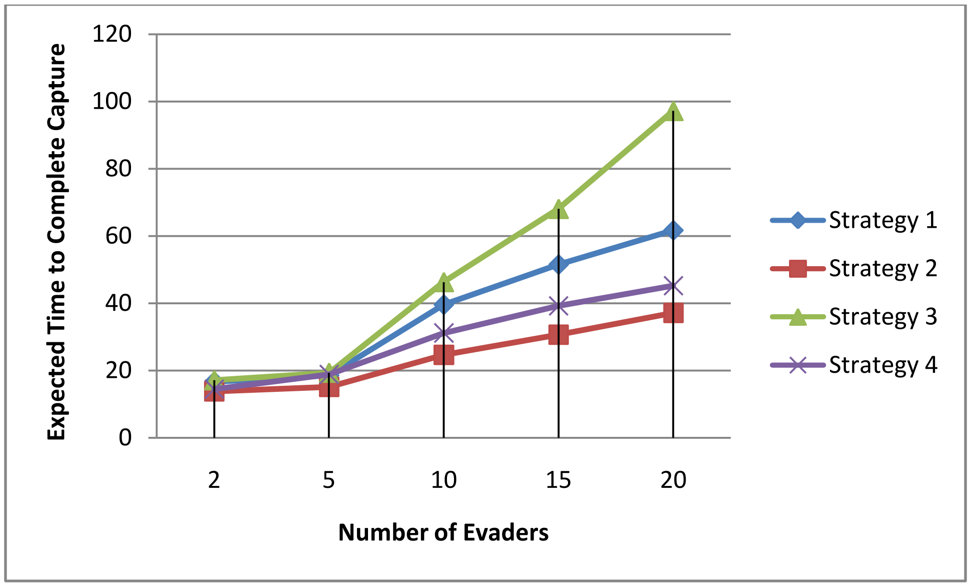


Figure 25. Given an initial distribution of the 20 evading targets in a circle around the pursuers, the average performance of the four evasion strategies after 1000 trial runs indicated that strategy 3 was significantly better than the others.

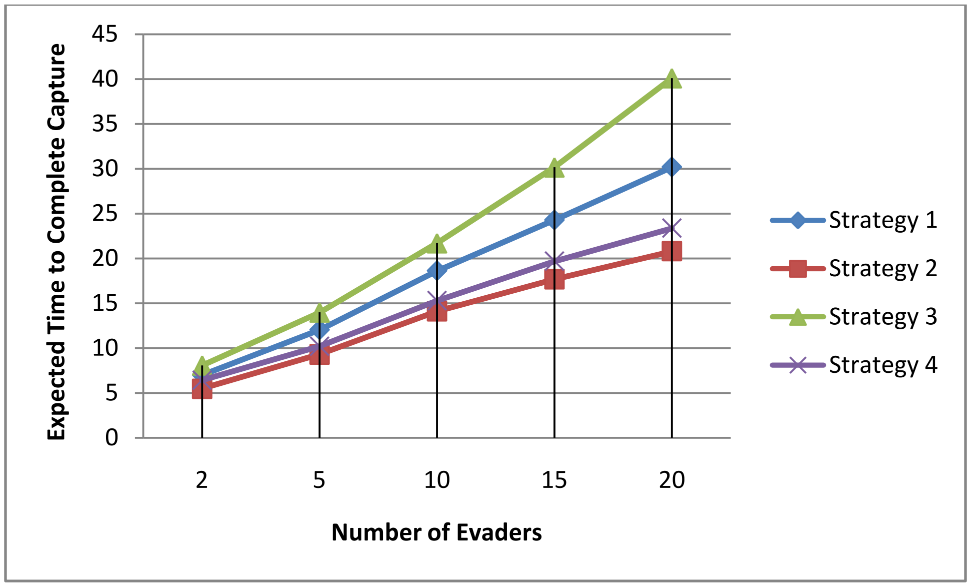
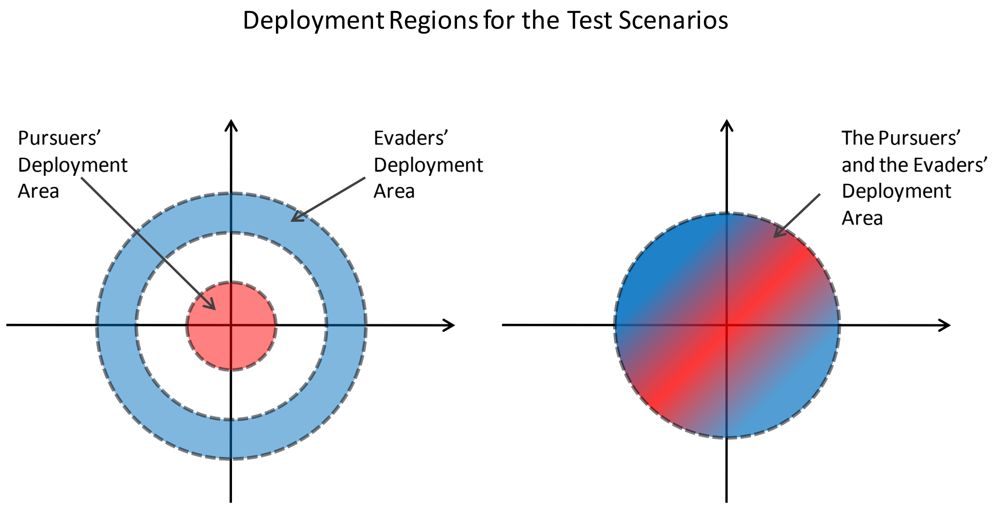


Figure 26. Given an initial distribution of the 20 evading targets in a circle with the pursuers, the average performance of the four evasion strategies after 1000 trial runs indicated that strategy 3 was significantly better than the others.



Deployment Regions for the Test Scenarios
Pursuers' Deployment Area
Evaders' Deployment Area
The Pursuers' and the Evaders' Deployment Area

Figure 27. Left: The evading targets are placed randomly placed in the region around the pursuers deployment areas. Right: The pursuers and the evaders are placed randomly in the same region.

4.1 Maximum Principle Formulation

Consider the vehicle dynamics

$$\dot{x} = f(x, u_p, u_e, t), \tag{28}$$

where *x* is the state vector of the game, up and ue are the controls of the pursuers and the evaders respectively, and *t* is time. The corresponding minimum-time Hamiltonian is given by,

$$\bar{H} = \min_{u_p} \max_{u_e} \langle \lambda, f(x, u_p, u_e, t) \rangle + 1$$

where $\langle , \rangle$ denotes the inner product. The evolution of the costates (λs) is given by the following differential equations:

$$\dot{\lambda} = -\left(\frac{\delta f}{\delta x}\right)^T \lambda \tag{29}$$

The optimality conditions that have to be satisfied for each of the control vectors up and ue are given by the following equalities:

$$\begin{gathered} \frac{\delta H}{\delta u_p} = 0, \\ \frac{\delta H}{\delta u_e} = 0 \end{gathered} \tag{30}$$

Furthermore, the following transversality condition must hold,

$$\bar{H}_{t=t_f} = 0, \tag{31}$$

where t_f is the termination time. To solve the system of *n* differential equations given by Equation 28 and 29, *n* boundary conditions must be provided. The players' known initial positions provide half of these conditions, and the game termination points along with Equation 30 provide the remaining conditions.

However, the approach cannot be implemented directly as described above. The trouble lies in the times at which the boundary conditions are prescribed. Since half of the boundary conditions are given at the termination time and the other half of the boundary conditions are given at the initial time, solving the system of differential equations is not simple. However, the shooting method can overcome this problem by propagating the solution forward multiple times and perform parameter correction to arrive at a stable solution.

Two very important assumptions are needed to ensure that Equations 28 through 31 are sufficient for an optimal solution, namely that *f(·)* is C^1 continuous in *x* and that the control domain is a convex body, that is, the control domain is convex and has a non-empty interior.

4.2 Constraint Considerations

In situations where constraints such as buildings, trees or even bounds on the control variables are introduced into the differential game, the Hamiltonian must be modified to accommodate the additional path constraints. Generally, the constraint will be given by the inequality,

$$C(x,t) \leq 0,$$

and consequently the expression for the Hamiltonian becomes:

$$\bar{H}_{constrained} = min_{u_p}\, max_{u_e} \langle \lambda, f(x, u_p, u_e, t)\rangle + \langle \lambda', C^q(x,t)\rangle + 1,$$

where $C^{\backslash q}(x,t)$ denotes the q^{th} time-derivative of *C(x,t).* The time derivative of the constraint is taken to ensure that the controls are represented explicitly in the constraint condition. For instance, if the *C(x,t)* expression must be differentiated three times to make the function an explicit function of the control variables, then $q = 3$.

If the constraint becomes active at any point, then $C^q(x,t) = 0$, $\lambda' > 0$ and $\dot{\lambda} = -\frac{\delta \bar{H}_{constrained}}{\delta x}$, otherwise $\lambda' = 0$ and $\dot{\lambda} = -\frac{\delta \bar{H}}{\delta x}$. In addition, upon entering and leaving a constrained region, the trajectory must enter along the tangent of the constraint, that is, $C(x,t) = 0$, $C^1(x,t) = 0$, …, $C^{q-1}(x,t) = 0$.

Since state constraints often introduce discontinuities in the derived optimal trajectories, it is important to maintain continuity at “corner” points, that is, points at which the control strategy changes significantly. Hence the following equalities must hold:

$$\begin{gathered}\lambda^T(t^-) = \lambda^T(t^+) + M^T \cdot N_x \\ \bar{H}(t^-) = \bar{H}(t^+) - M^T N_t \quad , \\ \bar{H}_u(t^-) = \bar{H}_u(t^+)\end{gathered}$$

where $N = (C(x,t), C^1(x,t), \ldots, C^{q-1}(x,t))^T$, M^T is a vector consisting of q constant Lagrange multipliers.

4.3 Two-player Game Example

To illustrate the approach, some numerical experiments are conducted using the following vehicle dynamics:

$$\begin{gathered}\dot{x}_p = V_p \cdot cos(\theta_p), \\ \dot{y}_p = V_p \cdot sin(\theta_p), \\ \dot{\theta}_p = \omega_p \\ \dot{V}_p = a_p - k_1 \cdot (V_p - \alpha_1)\end{gathered}$$

$$\begin{gathered}\dot{x}_e = V_e \cdot cos(\theta_e), \\ \dot{y}_e = V_e \cdot sin(\theta_e), \\ \dot{\theta}_e = \omega_e \\ \dot{V}_e = a_e - k_2 \cdot (V_e - \alpha_2)\end{gathered}$$

where *(ap, ωp)* and *(ae, ωe)* are the pursuing and the evading players control, *x*, *y*, *V*, and *θ* are the players’ position, velocity and heading states, k_1 and k_2 are the players’ drag coefficient, and α_1 and α_2 are positive constants that ensure that the velocity is always greater than zero.

The targets are considered captured if the pursuer is within a circle of radius four of the target. In the examples where there are multiple targets, the targets that have been captured are removed from the game, while the remaining players continue the game until all the targets have been captured.

The Hamiltonian used to derive the optimal control strategies are given by:

$$\bar{H} = \min_{a_p,\omega_p} \max_{a_e,\omega_e} \Big\{ \lambda_{x_p} \cdot V_p \cdot cos(\theta_p) + \lambda_{y_p} \cdot V_p \cdot sin(\theta_p) + \lambda_{\theta_p} \cdot \omega_p + \lambda_{V_p} \cdot (a_p - k_1 \cdot V_p)$$
$$+ \lambda_{x_e} \cdot V_e \cdot cos(\theta_e) + \lambda_{y_e} \cdot V_e \cdot sin(\theta_e) + \lambda_{\theta_e} \cdot \omega_e + \lambda_{V_e} \cdot (a_e - k_1 \cdot V_e)$$
$$+ \lambda'_{a_{pu}} \cdot (a_p - A_p) + \lambda'_{a_{pl}} \cdot (-a_p) + \lambda'_{a_{eu}} \cdot (a_e - A_e) + \lambda'_{a_{el}} \cdot (-a_e)$$
$$+ \lambda'_{\omega_{pu}} \cdot (\omega_p - W_p) + \lambda'_{\omega_{pl}} \cdot (-\omega_p - W_p) + \lambda'_{\omega_{eu}} \cdot (\omega_e - W_e) + \lambda'_{\omega_{el}}$$
$$\cdot (-\omega_e - W_e) \Big\} + 1$$

where (W_p, A_p, W_e, A_e) are the players control bounds.

From the Hamiltonian, the following optimality conditions can be derived:

$$\frac{\delta \bar{H}}{\delta a_p} = \lambda_{V_p} + \lambda'_{a_{pu}} - \lambda'_{a_{pl}} = 0$$
$$\frac{\delta \bar{H}}{\delta \omega_p} = \lambda_{\theta_p} + \lambda'_{\omega_{pu}} - \lambda'_{\omega_{pl}} = 0$$
$$\frac{\delta \bar{H}}{\delta a_p} = \lambda_{V_e} + \lambda'_{a_{eu}} - \lambda'_{a_{el}} = 0$$
$$\frac{\delta \bar{H}}{\delta \omega_p} = \lambda_{\theta_e} + \lambda'_{\omega_{eu}} - \lambda'_{\omega_{el}} = 0$$

The termination condition is given by:

$$\Psi(x_p, x_e, y_p, y_e) = (x_p - x_e)^2 + (y_p - y_e)^2 - r^2$$

Therefore at termination the costates become:

$$\lambda(T) = \frac{\partial \Psi}{\partial x}\bigg|_{t=T} = \begin{bmatrix} 2 \cdot (x_p(T) - x_e(T)) \\ 2 \cdot (y_p(T) - y_e(T)) \\ 0 \\ 0 \\ -2 \cdot (x_p(T) - x_e(T)) \\ -2 \cdot (y_p(T) - y_e(T)) \\ 0 \\ 0 \end{bmatrix}$$

Determining the time-derivatives of the multipliers is done by determining the change in the Hamiltonian with respect to the states:

$$\begin{aligned} \dot{\lambda}_{x_p} &= -\frac{\delta \bar{H}}{\delta x_p} = 0 \\ \dot{\lambda}_{y_p} &= -\frac{\delta \bar{H}}{\delta y_p} = 0 \\ \dot{\lambda}_{x_e} &= -\frac{\delta \bar{H}}{\delta x_e} = 0 \\ \dot{\lambda}_{y_e} &= -\frac{\delta \bar{H}}{\delta y_e} = 0 \end{aligned} \tag{32}$$

From Equation 32, we know that the costates do not change over time; hence, the mapping of the termination conditions to the initial conditions is simple.

The two player game has been reduced to a parameter optimization problem given two parameters $\lambda_{xp} \in [-2, 2]$ and $\lambda_{xe} \in [-2, 2]$, where the evading target attempts to increase capture time by varying λ_{xe}, and the pursuer attempts to minimize the capture time by varying λ_{xp}.

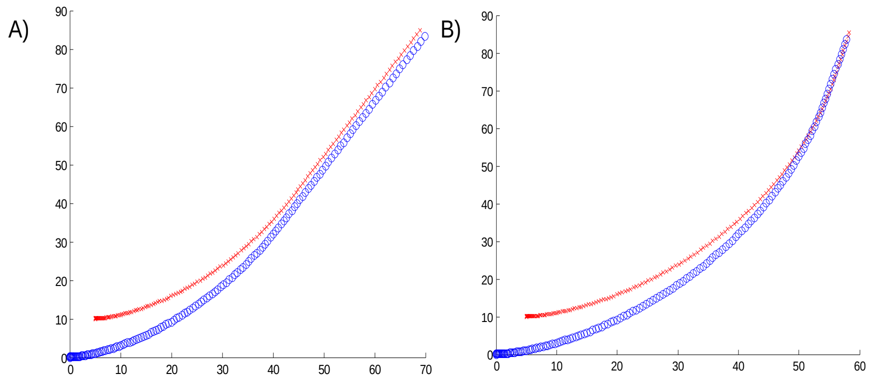


Figure 28. A) Trajectory with λ_{xp} = -1 and λ_{xe} = 1. Termination happened at t = 15.3 seconds., B) Trajectory with λ_{xp} = -.42 and λ_{xe} = .40. Termination happened at t = 15.0 seconds.

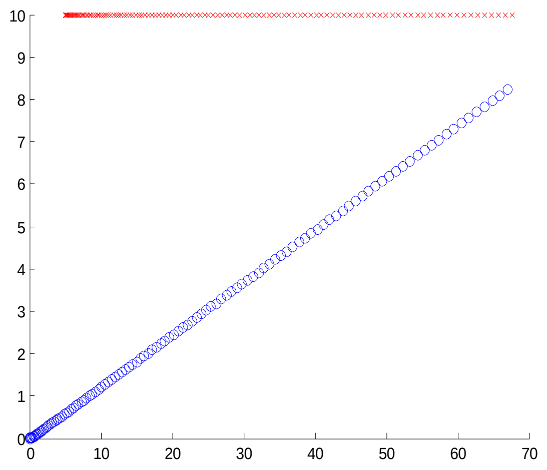


Figure 29. Trajectory with λ_{xp} = -1.88 and λ_{xe} = 2. Termination happened at t = 11.8 seconds.

In Figure 28 A), two simulations using different values of λ_{xp} and λ_{xe} are shown. The pursuer is performing suboptimally, since a simulation run where the pursuer intercepts the target directly from behind, reduced the interception time to 14.9 seconds. After several successive optimization steps were taken, the optimal interception and evasion strategy of the pursuer and the evader was found to result in the graph shown in Figure 28 B).

Since the above interception strategy was determined by optimizing over the costates, several suboptimal solutions had to be considered. One such attempt is shown in Figure 29. The purpose of this example is to show that it may be possible to guide the optimization of λ_{xp} and λ_{xe}, such that many poor costate choices are excluded. In the multi-player example, the values of the λs are updated while the simulation is executed, and likely initial values of the *λ's* are chosen based on interception observations. This enables the algorithm to run significantly faster since the costate search space is reduced significantly.

4.4 Multiplayer Game Example

The multiplayer game problem emphasizes the benefits derived from applying the maximum principle instead of relying on solving the Hamilton-Jacobi-Bellman-Isaacs (HJBI) equation. In this example, three pursuers must intercept three evading targets. The dynamics of each player are still described by the four-state model described in Section 1.3. Hence, even if the system is rewritten in relative coordinates, the value function derived from the HJBI equations can only be described completely in 20 dimensions, whereas the Maximum principle only requires that the value of six costates are correctly prescribed when the game is started.

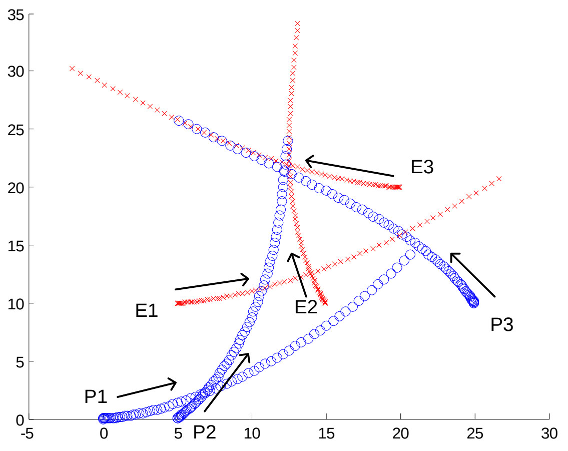


Figure 30. Multiplayer Game example with three pursuers intercepting three evading targets.

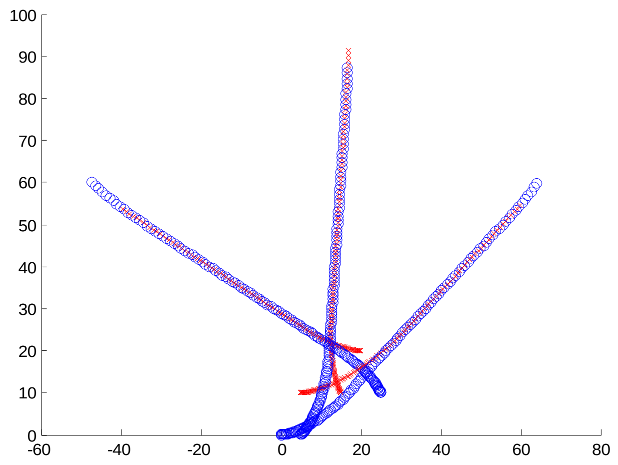


Figure 31. Termination of the game after 13.6 seconds.

In Figure 30 and Figure 31, a simple game consisting of six players is shown. In an attempt not to have to run the simulation multiple times, some heuristic rules were used to guide the optimization of the initial conditions on the costates. One of these rules is based on the behavior of the target right before capture. If the target is about to get captured by a pursuer it will use all its control authority to move away from the pursuer. Therefore, the target will attempt to move the line connecting the pursuer to the evader such that it will be pointing in the opposite direction of the evader's direction of travel.

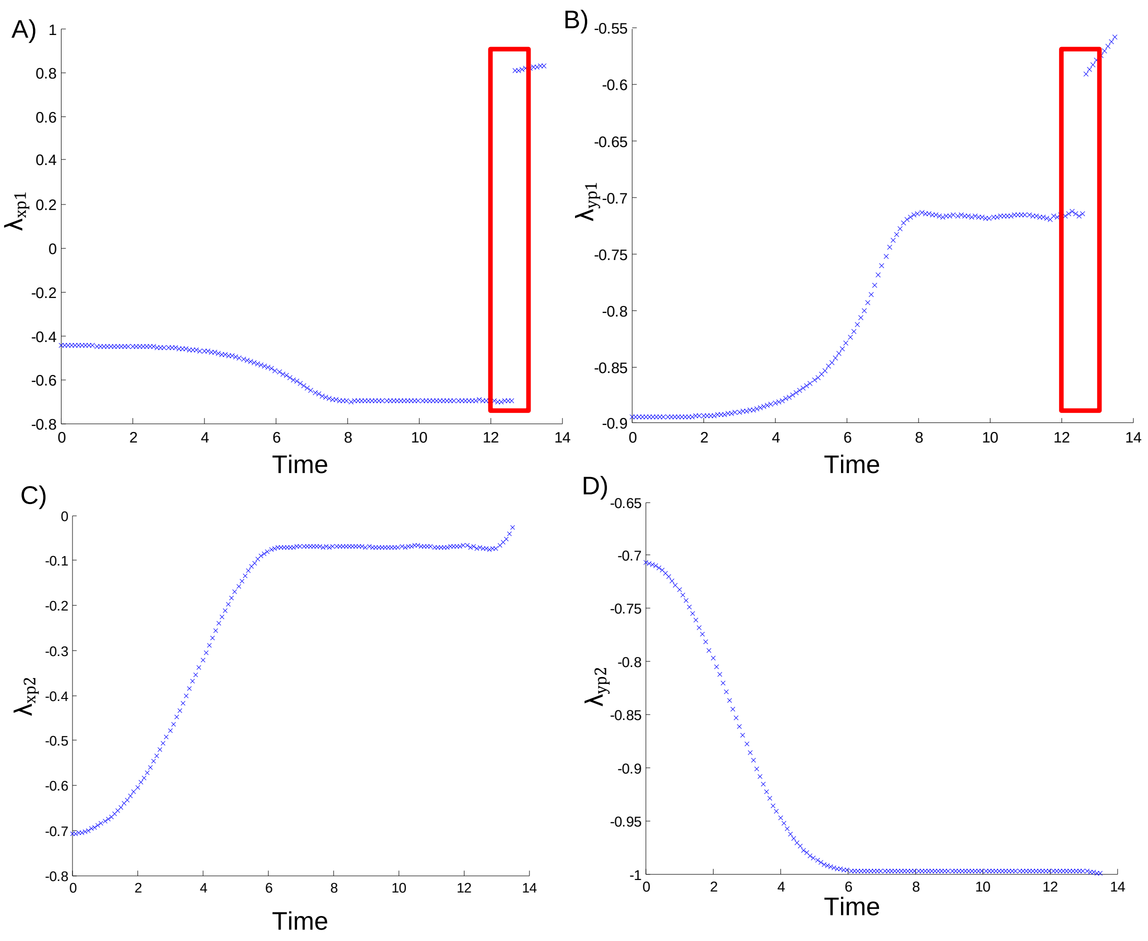


Figure 32. Evolution of the costate estimates.

As the test game is being played, the chosen costates are changed based on the evolution of the game. Hence, as the pursuers approach the evaders the multipliers become more and more accurate. Naturally, if an evading target is captured, the entire game is changed, and the costate estimates change abruptly. This change is shown in Figure 32 A and B. Since Pursuer 1 captures its target before Pursuer 2, a discontinuity occurs in the costate estimate indicated by the red boxes. The change does not affect Pursuer 2 greatly since the target that it is attempting to capture is still present; however, a small but much smoother change does occur as shown in Figure 32 C and D.

In Figure 33, the solution using the steady state values of the costate estimates are shown. Since the game essentially changes three times throughout the run, the costates should also be changed to account for the overall change in the objective. Consequently, once an evading target has been captured new values of the costates must be determined.

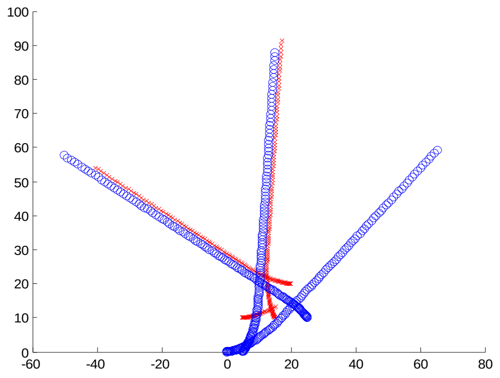


Figure 33. Simulation Example using the Steady State values of the Costates.

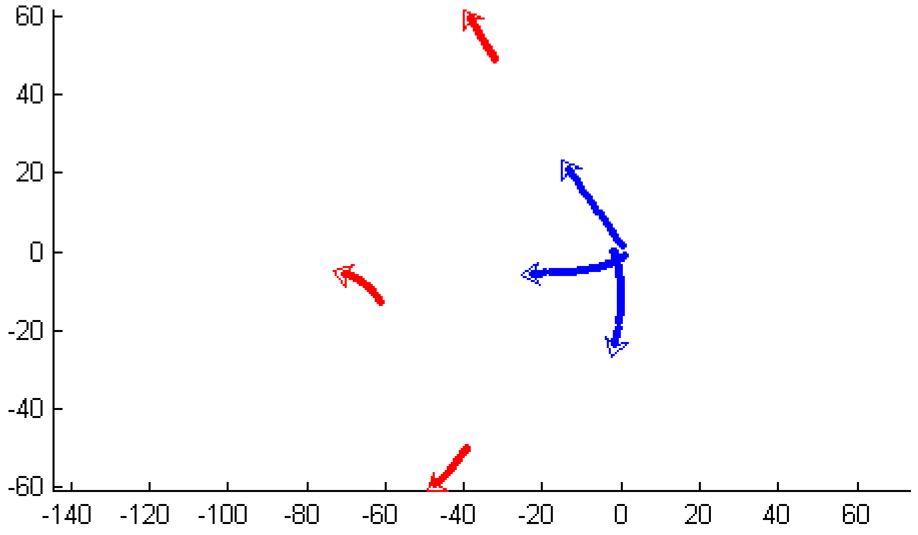


Figure 34. An example of a scenario in which there are more evading targets than pursuers. The position and heading of the pursuers is indicated by a blue triangle, while the position and heading of the evading targets are indicated by a red triangle.

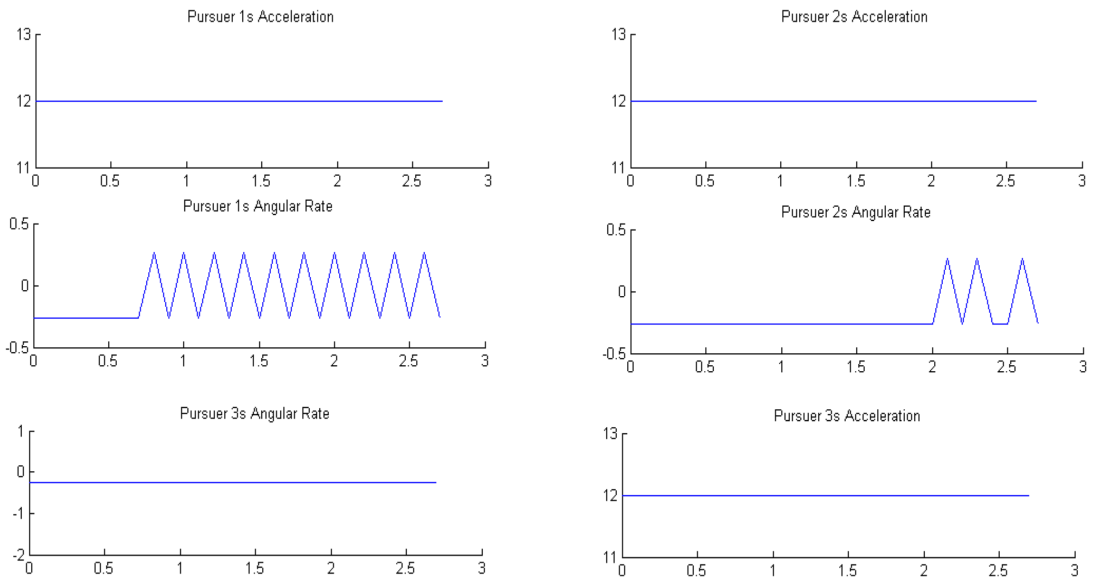


Figure 35. The control effort of the three pursuing players.

CHAPTER 5

PERFORMANCE MAP APPROACH

As mentioned in chapter 3 and chapter 4, the decomposition approach and the maximum principle approach require that solutions to the individual two-player differential games exist. Hence, it is assumed that the pursuers are slightly faster and more agile than the evading targets. However, one important reason for enabling multivehicle systems to cooperate is to allow these systems to accomplish tasks that the individual vehicle cannot perform on its own.

As a natural extension to the decomposition work, one might consider if it is possible to capture much faster targets if the number of pursuers is larger than the number of evaders. To address this problem, minimum time information from each pursuer and evader is used to derive containment and, if possible, interception strategies.

5.1 Evaluating the Players' Capabilities

Some information about the capabilities of the players must be obtained before a containment strategy can be derived. The computational burden associated with determining a solution using the classical differential game framework is too high in all but the simplest scenarios. Therefore, the standard minimum time Hamilton-Jacobi-Bellman-Isaacs equation,

$$min_{u_p \in U_p} max_{u_e \in U_e} \left\{ \langle \frac{\partial V}{\partial \vec{x}}, f\left(\vec{x}, \vec{u}_p, \vec{u}_e, t\right) \rangle \right\} + 1 = 0 \tag{33}$$

will be replaced by a collection of simpler Hamilton-Jacobi-Bellman (HJB) minimum time problems,

$$\begin{gathered}
min_{u_{p1}\in U_p}\left\{\langle\frac{\partial V_{p1}}{\partial \vec{x}_{p1}}, f_{p1}\big(\vec{x}_{p1},\vec{u}_{p1},t\big)\rangle\right\}+1=0 \\
\vdots \\
min_{u_{pn}\in U_p}\left\{\langle\frac{\partial V_{pn}}{\partial \vec{x}_{pn}}, f_{pn}\big(\vec{x}_{pn},\vec{u}_{pn},t\big)\rangle\right\}+1=0 \\
min_{u_{e1}\in U_e}\left\{\langle\frac{\partial V_{e1}}{\partial \vec{x}_{e1}}, f_{e1}(\vec{x}_{e1},\vec{u}_{e1},t)\rangle\right\}+1=0 \\
\vdots \\
min_{u_{m}\in U_e}\left\{\langle\frac{\partial V_{em}}{\partial \vec{x}_{em}}, f_{em}(\vec{x}_{em},\vec{u}_{em},t)\rangle\right\}+1=0
\end{gathered} \tag{34}$$

where, $u_p = \{u_{p1}, u_{p2}, ..., u_{pn}\}$ are the controls of the n pursuers, $u_e = \{u_{e1}, u_{e2}, ..., u_{em}\}$ are the controls of the m evaders, $x \in \mathrm{R}^{(n+m)\cdot s}$, s is the size of each of the players' state space, $x_{ei} \in \mathrm{R}^s$ for $i = 1, 2, \ldots m$, $\mathrm{x}_{\mathrm{pj}} \in \mathrm{R}^s$ for $j=1, 2, \ldots n$, and V_{ei} (and V_{pj}) is the minimum time it takes evader i (and pursuer j respectively) to reach a particular state. It should be noted that Equation 33 and the system of equations shown in Equation 34 are not generally equivalent, that is, by reducing the space over which the optimization is performed some information is lost.

Given the initial positions of the players and appropriately chosen termination conditions, it is possible to propagate each of the systems listed in Equation 34 forward using the Fast Marching Method presented in [33]. Solving Equation 34 instead of Equation 33 is computationally much simpler, since the space over which the optimization is performed is much smaller. For the general multiplayer game, the optimization space is a subset of $\mathrm{R}^{(n+m)\cdot s}$, while the systems shown in equation 30 are only optimized over a subset of R^s $(m+n)$ times.

Once the minimum time problems have been solved, the performance estimates are combined into a minimum time map. The map consists of $m+n$ regions, where each region describes a subset of the state space for which a particular player can reach any state before the other players get to that state. For the evading targets, each of these regions is considered safe. That is, none of the pursuers can intercept the evaders while they are moving within their region.

Containment and capture of a target is closely related to the targets' safe regions. If an evader has a safe region, it is guaranteed to be able to avoid capture for at least as long as the highest value within its region. Hence, if the largest value within evader *i*'s region is $V_{ei} = \tau$ seconds, then the evader can avoid capture for at least τ seconds. It is the primary objective of the pursuing players to ensure that all the evading players' combined regions do not become infinitely large, since this implies that containment of the players is not maintained. The secondary objective is to attempt to reduce size of the combined regions to zero which ensures that all the targets will be captured.

5.2 Strategy Considerations

To determine the general containment strategy, it is beneficial to consider certain scenarios for which Euclidian distance measures provide a good estimate of time to intercept. It should be noted, that the approach is not restricted to such situations, since the minimum time problems are solved directly from the HJB equations, but, from a conceptual perspective, the simple distance considerations aid in determining effective containment strategies.

Several strategies will be considered, however all of them will rely on the performance based feedback loop shown in Figure 36. The performance information will be used extensively in an attempt to construct an adaptable framework for containing and intercepting the targets.

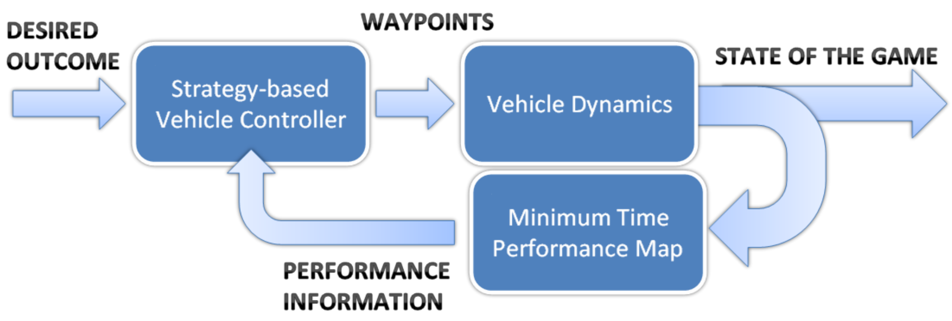


Figure 36: Closed Loop representation of The Containment Controller.

5.2.1 *The Simple Evasion Strategy*

Based on the derived minimum time regions, it is easy to determine if a simple evasion strategy is enough to ensure that an evading target can escape the pursuing players.

Definition 1: A Simple Evasion Strategy is a strategy determined at the beginning of the game resulting in a single predetermined trajectory which is not changed throughout the game.

The simple evasion strategy is surprisingly effective in containment problems. The reason for the simple strategy’s effectiveness resides in the fact that the pursuers cannot assume that the evaders are using such a strategy; hence, the pursuers will have to spend time on securing all the possible escape routes while the evaders can concentrate all their efforts on one particular escape corridor.

The simple escape strategy is guaranteed to work if the following axiom holds:

Axiom 1: If the minimum time region of an evading target is not enclosed by the minimum time regions of the pursuers then there exists a simple evasion strategy which ensures that the evader is not captured.

Equivalently, the pursuers cannot win a game for which their combined regions do not enclose all the evading targets. It should be noted that this condition is sufficient for the evaders but only necessary for the pursuers. Figure 37 shows a simulated example in which the evader is capable of escaping the pursuers using the simple escape strategy. The longer it takes a player to reach a particular state, the darker the corresponding color is.

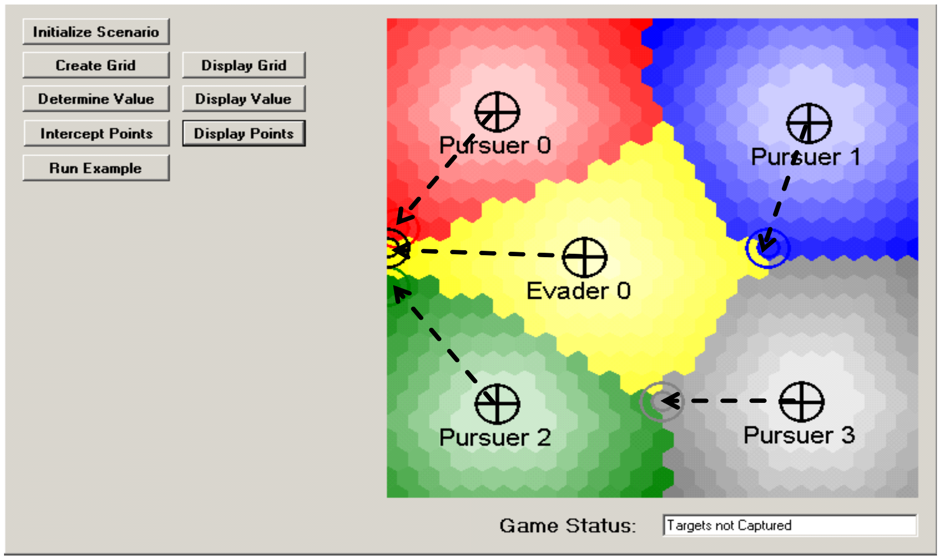


Figure 37. Depiction of a scenario in which the evading target is guaranteed to escape using a simple escape strategy. In this example, The Target is 40% faster than the pursuers.

5.2.2 Determination of Possible Escape Points

The simple evasion strategy hinges on the ability of the evader to determine the best possible escape corridors. If at all possible, the evading target should attempt to move toward points that it can reach before any of the pursuers; however, this requirement does not provide information about how close the evading target is to escaping the pursuers. In fact, the evader will attempt to move outside the enclosure formed by the pursuers.

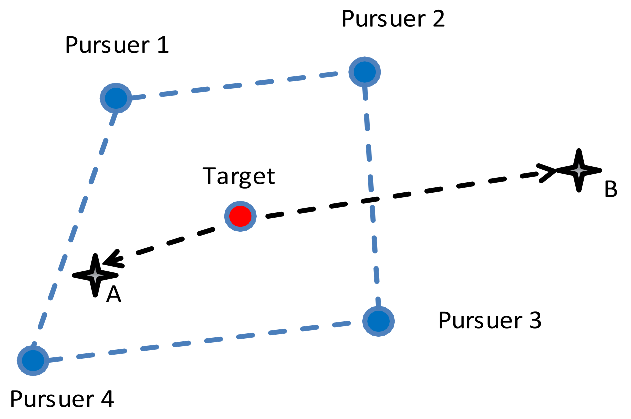


Figure 38. Two points 'A' and 'B' that the target can reach before the Pursuers.

As shown in Figure 38, it is advantageous for the target to move toward point B, since that point effectively eliminates Pursuer 1 and Pursuer 4 from the game. Hence, it is the objective of the evader to breach containment by maximizing the time to all the pursuers whose minimum time regions are adjacent to the minimum time region of the evading target.

Proposition 1: The targets will attempt to reach points that maximize the sum of the values of the subset of pursuers that are adjacent to the targets' minimum time regions. Hence, if X is the set of states that can be connected to the current state of the evader e_i by a trajectory 'C' such that x' on C, $V_{ei}(x') \leq min_{k=1,2,\dots,n}(V_{pk})$. Furthermore, let S_{pei} be the set of pursuers for which

$$S_{pei} = \{p_j \in \{p_1, p_2, \dots, p_n\}, x' \in X | V_{pj}(x') = V_{ei}(x') \leq min_{\,k=1,2,\dots,n}(V_{pk})\},$$

then the critical escape point satisfies:

$$x_{critical} = \left\{x' \in X | \textstyle\sum_{p \in S_{pei}} V_p(x') \geq max_{x'' \in X}\left(\sum_{p \in S_{pei}} V_p(x'')\right)\right\} \qquad (35)$$

The performance value of the critical point will be the corresponding sum of the minimum time values. It should be noted that the critical escape points are not unique which is advantageous for the evading target since the pursuers will have to handle all the critical points simultaneously.

5.2.3 *The Adaptive Pursuit and Evasion Strategies*

As the game progresses, avenues of opportunity may present themselves to the evaders. In such instances, the evaders should update their strategies based on the current state of the game. Hence, the determination of the critical escape points has to be done continually throughout the game, and since the critical points are dependent on the underlying minimum time problems, these problems will also have to be solved repeatedly throughout the game.

The pursuers must adopt a strategy that is dual to the evasion strategies, since the pursuers must tailor their containment strategy based only on the current state of the targets. Hence, even if the targets adopt the simple evasion strategy, the pursuers must ensure that all other escape routes are covered. Such a situation is depicted in Figure 39, in which the evader uses a simple escape strategy but the pursuers will have to ensure that all the best escape routes are covered.

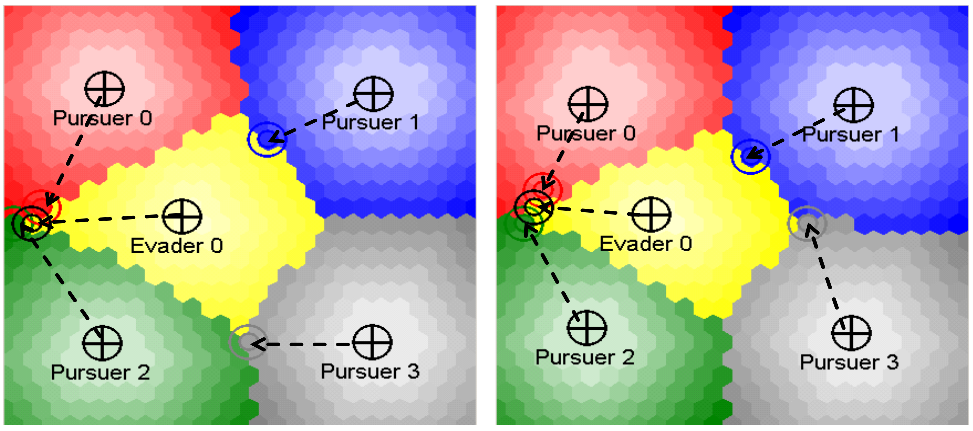


Figure 39. Left: Initially The Four Pursuers are cutting the left, the top and the bottom escape routes off. Right: later pursuer 3 ensures that the target does not escape through the right escape corridor. The evader was 20% faster in this example.

Ideally, the pursuers' strategies are efficient; however, due to the computational burden related to determining the underlying performance map, there is a reaction delay which the targets may be able to take advantage of. The evasion strategy that should be adopted to take advantage of the computational delay is quite simple: Pick the best escape route the pursuers are not attempting to block. The purpose of this strategy is to increase the performance estimate of the escape corridor such that it becomes more critical than the ones the pursuers are concerned with. Hence, the pursuers must use some resources (time) to close the new corridor.

Ultimately, such a strategy is only successful if the performance estimates of the different corridors are close to one another or if a particular pursuer is forced to handle two important escape corridors. A scenario depicting the latter is shown in Figure 40. Pursuer 2 is responsible for closing the two best escape corridors, and is therefore forced to let one of the evading targets escape.

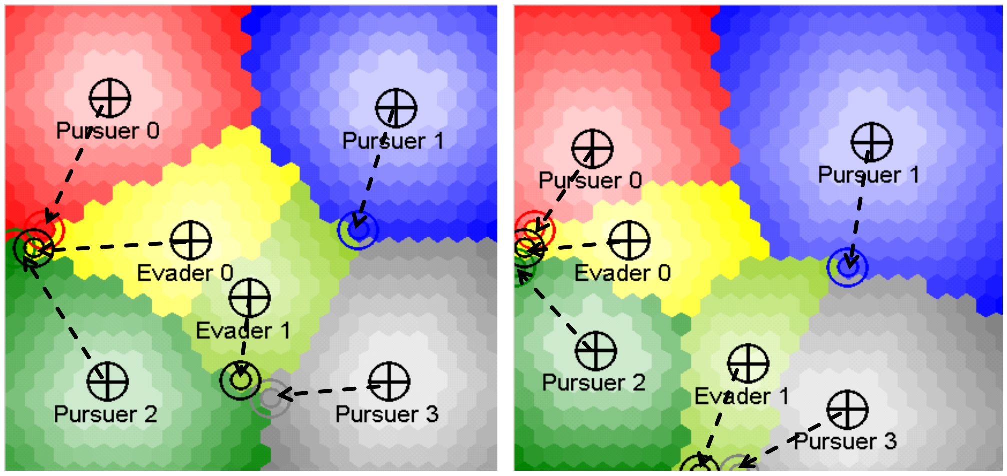


Figure 40. Left: initial Target Assignment forces Pursuer 2 to intercept Evader 0. Right: Evader 1 escapes through the corridor forced open by Evader 0. The two evaders are 20% faster

5.3 Target Capture Condition

Determining when it is possible to contain and intercept all the targets is nontrivial. In the previous section an example was shown in which the two evading targets initially appeared to be contained. However, due to the collaboration between the evading targets, one of the evaders was able to escape capture by forcing one of the pursuers to choose between the two targets.

Consequently, whether or not the targets can be intercepted is not only dependent on the individual player's capabilities, but also strongly dependent on the initial conditions. Hence, to ensure that a given problem can be solved, one must show that the value at the critical points discussed in proposition 1 is not only decreasing but also converges to zero. If this condition holds, containment and interception of the targets is guaranteed. On the other hand, if the convergence to zero cannot be shown but it can be shown that the value of the critical points is decreasing, then interception cannot be guaranteed but the targets will be contained. These conditions are not necessary but sufficient to show capture and/or containment.

5.4 Global Constraints

In many pursuit evasion problems, the environment can be used by the players to either outmaneuver the pursuers or to trap the evaders. Hence, when designing the adversarial strategies, it is important to consider the possibilities present in the environment. The interactions between the players and the environment can be incorporated into the underlying performance map. If a particular region is difficult to pass through, the velocity of the players can be reduced as they are passing through the region. The Fast

Marching Method can accommodate such constraints on the vehicle dynamics quite easily, as long as the effects can be accounted for using the vehicle states.

Impassable constraints such as buildings, water ways or very dense forest actually improve the performance of the algorithm, since the states that lie within the constraints will take infinitely long to reach, and will therefore not have to be determined when the underlying performance map is constructed. Figure 41 depicts a situation in which a barrier effectively prevents Pursuer 2 from participating in containing the target. From the evading target's perspective, Pursuer 2 simply does not exist, due to the dense forest separating the two. Since Pursuer 2's minimum time region is not adjacent to the evaders region, it is not assigned an interception point. It should be noted that in the case depicted in Figure 41, the evading target is not benefiting from the introduced barrier even though Pursuer 2 is effectively eliminated from the problem.

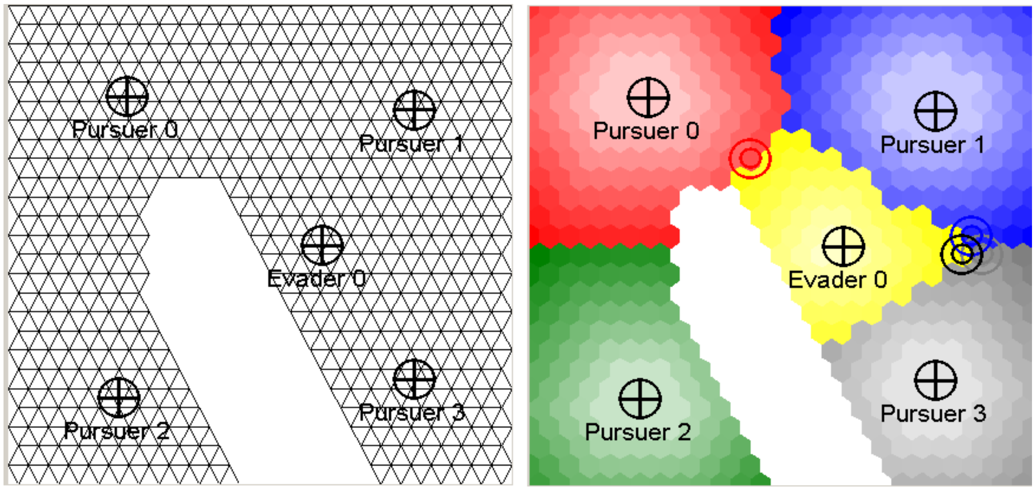


Figure 41: Left: The underlying grid with an inserted barrier. Right: the corresponding Performance Map. The target is 20% faster than the pursuers.

5.5 Algorithm Implementation

Since generating the underlying performance map must be done rapidly to insure that the containment is performed effectively, it is important to implement the Fast Marching

Method to run as efficiently as possible. The method used to generate the simulation results relies on the Ordered Upwind Method, which decouples the nonlinear systems by using information about the characteristic directions. The particular method used in this case was developed by J. Sethian and A. Vladimirsky [33].

The algorithm is implemented over a triangular mesh stored in a linked list. Since the propagation of the solution is performed for each player, the grid is constructed such that each point can contain the information about all the players. This significantly decreases the data management overhead.

The value update is determined based on the following equation:

$$V(x) = min\left\{V(x), min_{x_i,x_j \in NS(x)} V_{x_i,x_j}(x)\right\} \tag{36}$$

Where

$$V_{x_i,x_j}(x) = min_{\alpha\in[0,1]}\left\{\frac{\|\alpha\cdot x_i+(1-\alpha)\cdot x_j-x\|}{f(x,v_\alpha)} + \alpha\cdot V(x_i) + (1-\alpha)\cdot V(x_j)\right\} \tag{37}$$

where $v_\alpha = \frac{\alpha\cdot x_i+(1-\alpha)\cdot x_j-x}{\|\alpha\cdot x_i+(1-\alpha)\cdot x_j-x\|}$ and *NS(x)* are the points in the neighborhood of *x*. A graphical representation of a single update step is shown in Figure 42. *f(x,v$_\alpha$)* is the speed associated with moving from the point on the line connecting x_i to x_j to the point *x*. As shown in Equations 36 and 37, the minimum value is found by considering all the combinations of vertices in the neighboring set of *x*, and interpolating between these points to improve the performance estimate.

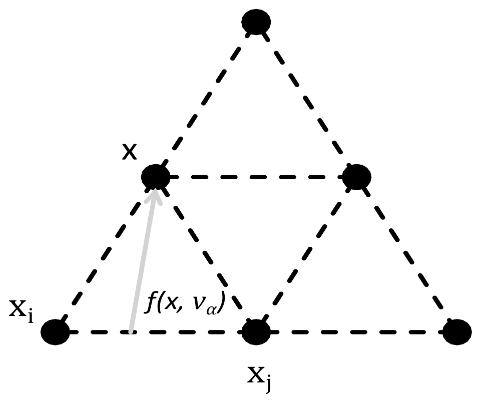


Figure 42: Example of updating the point 'x' based on a convex combination of the points x_i and x_j.

The outline of the algorithm is as follows:

1. The points in the mesh are marked as *Far* for each player, and the corresponding value is set to a maximum value.
2. The points inside the termination manifolds are set to *Accepted* for the corresponding player.
3. The points adjacent to the *Accepted* points are marked *Considered*, and the points' values are updated using the update rule described above. References to the *Considered* points are then added a *Considered* list.

The following steps are repeated until the value of all the points in the grid has been determined, that is, as long as the *Considered* list is not empty:

4. Find the point with the smallest value in the considered list, mark it *Accepted* and delete it from the *Considered* list.
5. Add the *Far* points that are adjacent to the removed point to the *Considered* list.
6. Update value of the *Considered'*points that are adjacent to the removed point.

Even though this algorithm is very efficient, it should still be noted that for real-time applications it is important to work with vehicle dynamics that can be described using as few states as possible.

5.6 Simulation Results

The simulations used to verify the players' strategies were based on the following simple vehicle dynamics,

$$\frac{d\vec{r}}{dt} \leq v_{player} \tag{38}$$

where v_{player} is the maximum velocity of a given player which the vehicle can move in any direction. The simple dynamic model ensures that the state space the optimization is performed over remains small enough to maintain fast execution of the algorithm.

As discussed earlier, the strategies were tested in several different scenarios, involving four pursuers and up to two evaders. The pursuers' initial positions were kept constant and are summarized in Table 2.

Table 2. Initial Positions of the Pursuers.

Pursuer Number	$x_{initial}$	$y_{initial}$
0	10	10
1	40	10
2	10	40
3	40	40

The pursuers' velocity was kept constant at unit velocity, and a target was considered captured if it was within 5 units of any of the pursuers. The grid resolution was set to 2 units. Table 3 summarizes the single vehicle results obtained by using the adaptive strategy.

Table 3.Results for a Single Target using adaptive escape strategy.

Target's Velocity	$x_{initial}$	$y_{initial}$	Interception Time
1.20	20	25	13 Time Units
1.30	20	25	14 Time Units
1.40	20	25	Containment Breached
1.40	25	25	15 Time Units

The interception times are not the maximum times that the evading target could obtain. The target could accept being contained, and by doing so extend the time to intercept slightly.

Table 4. Results for Two faster evading targets using an adaptive Escape strategy.

Targets' Velocities	$x_{initial,\ target\ 0}$	$y_{initial,\ target\ 0}$	$x_{initial,\ target\ 1}$	$y_{initial,\ target\ 1}$	Interception Time
1.10	20	25	25	30	Containment Breached
1.10	25	25	25	25	12 Time Units
1.20	25	25	25	25	22 Time Units
1.25	25	25	25	25	Containment Breached

As shown in Table 4, the targets' velocities can be reduced significantly while still enabling them to escape capture. In the instances where the targets escaped, the solution relied on the fact that one pursuer is unable to cover two different escape corridors. This behavior is not explicitly implemented into the targets' escape strategy, however as the scenario unfolds, the evading targets quickly identify such weaknesses and exploit them.

5.7 Additional Experiments

The approach used to capture multiple evading targets could possibly be extended to other scenarios. For instance, in the above analysis, the number of evading targets was assumed to be less than the number of pursuers. However, if the speed of the evading targets was decreased it may be possible for a set of pursuers to contain a much larger collection of evading targets. In such a case, the decomposition technique suggested in chapter 3 would be applicable since it is possible to obtain solutions to the two-player differential games. However, it is advantageous in many situations to have the pursuers contained before attempting to intercept them. Consequently, the pursuers should adopt the containment approach while the targets are enclosed by the pursuers, and switch to the two-player differential game approach if the targets are able to breach containment.

Figure 43 shows an example of four pursuers attempting to contain five evading targets. The targets are slower than the pursuers in this example; however, they are still able to breach containment. Once containment is breached, the pursuers should change their strategy to intercept the targets directly using the two-player differential game approach.

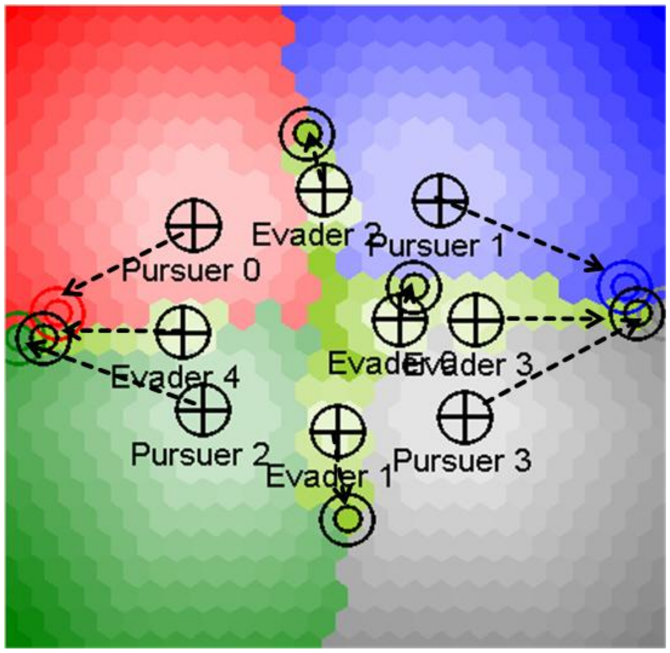


Figure 43. An Example with four pursuers and five evaders where The evaders' maximum velocity is 85% of the pursuers'.

Table 5 summarizes several numerical experiments performed on the scenario involving four pursuers and five evaders. The pursuers' initial positions were kept at the positions listed in Table 2, while all the evading targets started out at: (x, y) = (25, 25).

Table 5. Numerical experiments with four pursuers and five slow evaders.

Evaders' Velocity Relative to the Pursuers'	Time to Capture
50%	17 Time Units
55%	16 Time Units
60%	17 Time Units
65%	19 Time Units
70%	21 Time Units
71%	22 Time Units
72%	Containment Breached

Containment was considered breached if the targets were able to reach the edge of the considered area before the pursuers could intercept them. If containment is breached, it is likely that multiple targets are able to escape since the primary objective of the pursuers is to keep all of the targets contained. Hence, if a single target is breaching containment the pursuers will utilize all the available resources to reestablish containment.

CHAPTER 6

ADDITIONAL IMPLEMENTATION NOTES

Using the dynamic programming principle to solve the individual two-player differential games is unfortunately still a problem that remains exponentially complex with respect to the size of the individual player's state space. Hence, in order to solve these problems rapidly, special care must be taken to reduce the space over which the problem is solved and optimize the solution process. Reducing the optimization space is dependent on the specific problem considered, that is, if the two-player games are played in three dimensions, it may be possible to reduce the problems to ones played in the plane. In the cases where it is not possible to solve the differential game problems fast enough, researchers have utilized a limited look ahead approach [42], that is, only attempt to optimize the players' strategies over a smaller but computationally feasible planning horizon. However, for many games such an approach does not provide consistent strategies, since the system is not brought all the way to termination.

In an attempt to optimize the solution process, the underlying grid over which the Hamilton-Jacobi-Bellman-Isaacs equation is propagated will be optimized. A couple of methods to construct the regular grids will be discussed followed by an approach to reducing the overall computational complexity by reducing the number of vertices in the grid. Finally, the pseudo code of the algorithms used in the simulation environment will be provided along with some important implementation considerations.

6.1 Constructing the Grid

A grid is a collection of vertices and edges. The vertices contain the information required to construct the solution to the partial differential equation, and the edges describe which vertices are adjacent to one another, or equivalently, which of the neighboring vertices a given vertex can rely on to determine the solution.

For simple applications, it is very tempting to construct grids such that some of their structure is captured in the container in which they are placed. For instance, if a two-dimensional grid has to be constructed, one might initially use a two dimensional array to represent the grid, that is, the elements in the array contain the vertex information while the relative placement in the array describes the edges. Representing a grid in this manner is very intuitive and utilizes the system memory efficiently. However, if the number of edges is very large it is necessary to implement additional logic to determine which of the vertices are adjacent.

To address more exotic grids, a linked list approach can be adopted. The elements in a linked list contain the vertex and the edge information, that is, each element in the list keeps track of its neighbors, and it is therefore possible to construct any regular and irregular grids. However, since each element has to contain more information, it requires additional memory to represent a given grid, and it takes more processing time to initialize the grid.

With the additional computational and memory requirements the linked list approach may not seem to be a very attractive alternative. However, when grid refinement is discussed in the next section the advantage of the linked lists becomes apparent.

6.2 Grid Refinement

In some scenarios, the players in the game may have different constraints imposed on them by the environment. For instance, if a UAV is asked to intercept a ground target, the ground target could be constrained by buildings, trees or large bodies of water, while the UAV remain unconstrained by passing over the obstacles. Hence, the speed at which the solution to the Hamilton-Jacobi-Bellman-Isaacs equation propagates forward is dependent on those constraints. By exploiting this feature it is possible to optimize the underlying grid used to represent the solution to the problem, and thereby increase the execution speed of the algorithm.

If the evading target is constrained, the parts of the grid that is not within the constraints should be of higher resolution especially if the target has to maneuver through narrow corridors. In scenarios where the pursuers are constrained by the environment, a solution to the game may not exist. For instance, if a ground target is attempting to capture a UAV, the UAV will either remain within a constraint that the pursuer cannot enter or move such that the pursuer will have to negotiate the obstacles repeatedly, thereby ensuring that the ground vehicle is unable to capture the UAV.

The idea behind grid refinement is to reduce the grid resolution in certain parts of the state space while maintaining the required resolution in other parts. There two classes of grid refinement algorithms that can be adopted to refine grids. The first class is one in which a coarse grid is provided, and then additional points are added into the state space in the places where a higher resolution is required. The second class is essentially opposite to the first in that it assumes that a sufficiently fine grid is superimposed on the state space, and then vertices are removed from the parts that do not require the high resolution. The algorithm outlined here will be of the latter class for reasons which will be made clear in the next section. For other grid refinement approaches see [46].

6.3 The Refinement Algorithm

The algorithm consists of several non-trivial steps, and care should be taken to ensure rapid execution of each of them. The steps are:

1. Construct a linked list of the highest resolution required in any region of the state space.
2. If a vertex is in a region which does not require high resolution, mark the vertex as a potential “candidate” for reduction, otherwise mark the vertex as “fixed”.

Repeat the following until there are no “candidates” left:

3. Pick a “candidate” vertex, say vertex “A”.
4. If the vertex has no neighbors that are also “candidates” then go to step 9.
5. Pick a neighboring “candidate” vertex “B”.
6. Determine the largest distance from “A” to its neighbors (DA) and the largest distance from “B” to its neighbors (DB), and compute the quantities DA+DB/2 and DB+DA/2.
7. If DA+DB/2 and DB+DA/2 are less than the required resolution, copy the neighborhood of vertex “B” to vertex “A”, reposition vertex “A” between the two vertices, announce to “A”’s new neighbors that “A” is their neighbor, and delete vertex “B”.
8. Go to step 5 until vertex “A” has no other “candidate” neighbors that it can merge with.
9. Mark vertex “A” as fixed and go to Step 3.

When implementing the above algorithm, step 7 must be executed carefully, or an infinite loop may be constructed. Vertex “B” contains a reference to the neighboring vertex “A”, and unless it is removed the algorithm will get stuck in an infinite loop or “A” may simply disappear depending on how the linked list is implemented.

Insert Figures of a grid before refinement and one after.

An example of the grid refinement is shown in Figure 44. Notice that the number of neighbors that are adjacent to the vertices in the low resolution parts of the grid has increased significantly. This is a consequence of the interface between the high resolution areas and the low resolution areas. It is mainly due to this fact that it was decided to construct a high resolution grid and then reduce the number of vertices instead of constructing a coarse grid and improve the resolution in the areas that required it.

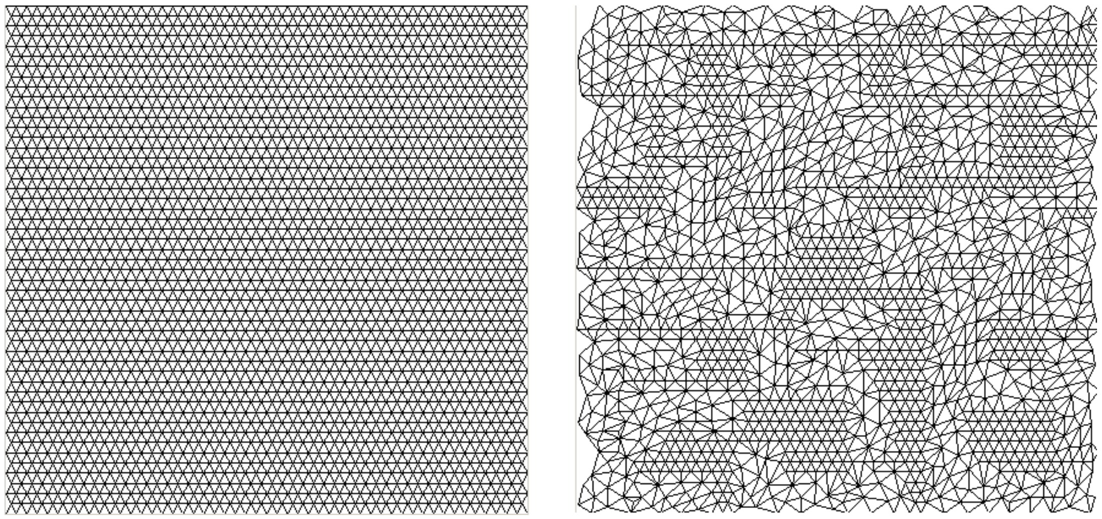


Figure 44. Left: Original grid before vertex reduction. Right: Same grid after vertex reduction.

When the high resolution regular grid is constructed it is simple to determine the neighbors of each vertex in the grid, since the vertices are a predetermined distance from one another. However, if a vertex has to be added to a particular region in order to improve the overall resolution, it becomes somewhat tedious and computationally expensive to determine its neighbors. Hence, by constructing a fine grid and then reduce the resolution, memory is sacrificed to improve execution speed.

6.4 Practical Implementation Limitations

As mentioned in section 6.2, the number of sample points in the state space greatly impacts the speed at which the solution can be generated. Hence, if the vehicles' dynamic models are represented by a large number of states, the computational burden may become too great to solve the problem quickly, simply due to the number of sample points in the state space. Consequently, it is important to attempt to simplify the vehicle dynamics as much as possible before attempting to solve the problem using the Decomposition Approach or the Performance Map Approach. Since the state space must be limited to only a few number of states, the Decomposition and the Performance Map approaches cannot be applied to problems where the dynamics can *only* be described in a very high dimensional space. Additionally, if the problem formulation is not continuous and at least once differentiable, the Maximum Principle approach cannot be applied either. In such cases, it may be necessary to rely on algorithms that are not as tightly coupled with the underlying dynamics.

Practically speaking, it is very difficult to determine good strategies for systems that have either very limited computational resources or are very complex in nature, so techniques that decouple the adversarial problem from the low-level dynamics are currently the only option in such situations.

CHAPTER 7

CONCLUSION AND POSSIBLE FUTURE WORK

With the increased utilization of UAVs to perform intelligence, surveillance, and reconnaissance missions, a demand for algorithms that will effectively manage and utilize the cooperative capabilities of multiple UAVs have emerged. The work described in this thesis addresses this need by providing three complimentary differential game based approaches to solving the multivehicle interception problem. The three approaches all rely on optimal control principles to evaluate the capabilities of the cooperating UAVs; however, the underlying assumptions that the three approaches are built around are all different.

The *decomposition approach* solves multiple two-player differential games using the dynamic programming framework, and then combines the resulting value functions in an attempt to approximate the solution to the complete game. The underlying assumptions that must be satisfied for this approach to be effective are that the pursuers are faster and more agile than the evading target, that the dynamics of each individual vehicle can be accurately described in a low-dimensional state space, and that the resulting two-player value functions are continuous. The advantage of decomposing the multiplayer game into multiple two player games is that the exponential complexity with respect to the number of players is effectively removed. However, the exponential complexity with respect to the size of the individual vehicle’s state space is still present.

The *maximum principle approach* mirrors the decomposition approach in that the complete multiplayer game is decomposed into several two-player games. The main difference between the two approaches is that the maximum principle approach only determines a single optimal interception trajectory while the dynamic programming approach determines the solution to an entire family of problems. Consequently, the maximum principle simplifies overall computational requirements needed to find a solution by computing the solution of the individual two-player games quickly. However, this approach can only be used if, in addition to the assumptions made for the decomposition approach, the problem description is sufficiently smooth.

Finally, the *performance map approach* addresses the problem where the two-player games cannot be solved, that is, the evading targets are either faster or more agile then the pursuing vehicles. Instead of capturing the vehicles' capabilities by solving multiple two-player games, several minimum-time optimal control problems are solved using the dynamic programming principle. By constructing a composite performance map, it is possible to determine where the evading targets are likely to attempt to escape. The pursuers then attempt to close these *escape corridors* in the fastest possible manner, that is, the primary objective of the pursuing vehicles is to contain the targets and the secondary objective is to intercept the targets. One important difference between this approach and the decomposition and maximum principle approaches is that capture of the evading targets is not guaranteed.

Several simulations is performed to verify the performance of the three multivehicle interception algorithms. However, since the complete multivehicle game solution is not actually computed, the escape strategies of the evading targets had to be approximated. For the scenarios used to test the decomposition and maximum principle algorithms, it is shown that the most effective escape strategy was *not* to simply rely on the value function of the nearest pursuer. For instance, if the targets that are not being targeted directly moved away from the nearest targeted evaders, the overall cost of intercepting all of the targets increased significantly. However, for simplicity the escape strategy used by the evading targets were simply to increase the expected cost of interception with respect to the nearest pursuer.

The evasion strategy used to test the effectiveness of the performance map approach was much simpler. While a given scenario is being tested, the evaders determine the best possible escape corridors by leveraging the minimum-time information determined by solving the multiple minimum-time problems. Since the pursuers use a fixed interception strategy, it is simple to determine the best possible escape strategy for the evaders.

7.1 Contributions

The work presented in this thesis contributes to the field by providing:

- Fast decision algorithms which enables multiple cooperating vehicles to engage multiple targets efficiently by,
    - Decomposing the complete multivehicle problem into smaller and more manageable two-player problems.
    - Effectively assigning tasks to the individual pursuing vehicles based on the derived performance estimates.

- An effective approach to managing multiple unmanned systems in collaborative engagement scenarios.

Several simulations were conducted to verify the performance of the suggested approaches.

7.2 Possible Future Work

To implement the three collaborative engagement algorithms on current UAV platforms, several issues must be addressed. The complete information assumption made when deriving the interception algorithms is a very difficult satisfy. First, the problem of locating a target, estimating its current state, and determining its dynamic model using the sensor suite onboard the UAVs must be solved. These three topics are in themselves very challenging and are all active research topics in robotics. Additionally, to utilize the computational power of the entire team of UAVs it is, as indicated in chapter 3, desirable to implement the cooperative engagement algorithms in a distributed fashion. However, transmitting performance information over a network of UAVs can be very costly, especially when comparing performance information over the entire state space is needed.

To limit the amount of data transmitted between the vehicles, it may be useful to consider the structure of wireless ad-hoc networks. Broadcast trees have been used to route communication in ad-hoc wireless networks [47, 48]. Such tree structures could potentially provide the necessary structure to transmit the two-player value functions over a network. Each parent node could merge the performance data received from its children, and then only pass on the combined performance data to its parent node. Not only will such an algorithm limit the amount of information transmitted over the network, but since the children of a parent node commonly are close to one another, it is likely that they are assigned to the same target.

APPENDIX A

OPTIMAL CONTROL PRINCIPLES

Optimal control theory is the branch of general control theory that attempts to derive the best possible controllers given a certain performance metric. Consequently, optimal control problems consist of a set of constraints and a performance metric.

Optimal control theory is an extensive field of study, which relies equally on engineering and mathematical insight. The purpose of this appendix is to provide the mathematical foundation to understand the differential game concepts introduced throughout the thesis. The two approaches to solving optimal control problems highlighted in this appendix; namely, the Dynamic Programming and the Maximum Principle approaches, are not meant to be an exhaustive introduction to optimal control. However these two techniques are used extensively in optimal control problems and used almost exclusively in the work presented in this thesis.

It is assumed throughout this short introduction, that the system to be controlled is given by a set of differential equations of the form,

$$\dot{\vec{x}} = f(\vec{x}, \vec{u}, t) \tag{39}$$

where $\vec{x}$ is the current state of the system, $\vec{u}$ is the control and *t* is time.

The performance metric or cost functional will be given by,

$$\min_{\vec{u}} J = \varphi\big(x(T)\big) + \int_{t_0}^{T} L(\vec{x}, \vec{u}, t)\, d\tau, \tag{40}$$

where *L()* is the integral cost function, *φ()* is the final state cost, *T* is the termination time, and t_0 is the initial time. Since most of the focus in this thesis is placed on minimum time problems, additional emphasis is placed on the minimum time versions of the Dynamic Programming and the Maximum Principle approaches. The cost functional for such problems is given by,

$$\min_{\vec{u}} J = \int_{t_0}^{T} d\tau = T - t_0. \tag{41}$$

Dynamic Programming Approach

Dynamic programming is a technique used to describe an entire family of optimal control problems, that is, given a system model the dynamic programming technique determines the optimal control solutions for a set of initial conditions. The relationship between the families of solutions is given by the Hamilton-Jacobi-Bellman equation which is a non-linear first-order differential equation.

The solution to the family of problems is represented by a value function, that is,

$$V(\vec{x}, t) = \min_{\vec{u}} J\,(\vec{x}, t, \vec{u}),$$

where $\vec{u}$ is the control minimizing the cost function *J*, *x* is the state of the system and *t* is time.

Before arriving at the Hamilton-Jacobi-Bellman equation, it is important to understand Bellman's principle of optimality.

Bellman's Principle of Optimality:

Given any pair $(t_a, x_a) \in [t_0, T]$,

$$V(t_a, \vec{x}_a) = inf_{\vec{u} \in U}\left\{\int_{t_a}^{t_b} L(\vec{x}, \vec{u}, t)\, d\tau + V(t_b, \vec{x}_b)\right\}, \forall t_0 \leq t_a \leq t_b \leq T.$$

where *U* is the set of admissible controls for which $\vec{u}$ is measureable. In other words, the value of a particular state at an earlier time can be determined by utilizing the values of the states at a later time, or equivalently, the last part of an optimal trajectory is optimal.

Two important assumptions are made when proving Bellman's principle of optimality, and those are:

1. (U, d) is a separable metric space.
2. *f(), L(),* and $\varphi()$ are uniformly continous, and there $\exists k \in \mathbb{R}$, $k > 0$ such that for $\alpha(\vec{x}, \vec{u}, t) = f(\vec{x}, \vec{u}, t)$, $\alpha(\vec{x}, \vec{u}, t) = L(\vec{x}, \vec{u}, t)$ or $\alpha(\vec{x}, \vec{u}, t) = \varphi(\vec{x})$

$$\begin{cases} \left|\alpha(\vec{x},\vec{u},t)-\alpha\left(\hat{\vec{x}},\vec{u},t\right)\right| \le k\cdot\left|\vec{x}-\hat{\vec{x}}\right|, \forall t\in[t_0,T], \vec{x},\hat{\vec{x}}\in\mathbb{R}^n, \vec{u}\in U \\ \qquad\qquad |\alpha(0,\vec{u},t)| \le k, \forall(t,\vec{u})\in[t_0,T]\times U. \end{cases}$$

These two assumptions must be verified before attempting to apply Bellman's principle of optimality.

In an attempt to simplify the determination of the continuously differentiable $C^1([t_0,T]\times\mathbb{R}^n)$ value function *V()*, the following partial differential equation along with a set of termination conditions can be used to generate the value function:

$$\begin{cases} \frac{\partial v}{\partial t} + sup_{\vec{u}\in U}\{\langle\vec{\lambda}, f(\vec{x},\vec{u},t)\rangle + L(\vec{x},\vec{u},t)\} = 0 \\ \qquad v|_{t=T} = \varphi\left(\vec{x}(T)\right) \end{cases}, \left(t,\vec{x},\vec{u},\vec{\lambda}\right)\in[t_0,T]\times\mathbb{R}^n\times U\times\mathbb{R}^n \quad (42)$$

where $\vec{\lambda} = \left[-\frac{\partial v}{\partial x_1}, -\frac{\partial v}{\partial x_2}, \cdots, -\frac{\partial v}{\partial x_n}\right]$, and $\langle\vec{\lambda}, f(\vec{x},\vec{u},t)\rangle$ denotes the inner-product of $\vec{\lambda}$ and $f(\vec{x},\vec{u},t)$. Equation 42 is commonly referred to as the Hamilton-Jacobi-Bellman equation, and $\langle\vec{\lambda}, f(\vec{x},\vec{u},t)\rangle + L(\vec{x},\vec{u},t)$ is commonly referred to as the Hamiltonian and is abbreviated by $H\left(t,\vec{x},\vec{u},\vec{\lambda}\right)$.

Solving an optimal control problem using the dynamic programming approach and a set of initial conditions consists of three steps:

1. Determine the value function by solving the Hamilton-Jacobi-Bellman equation.
2. Given the value function, determine the control that maximizes the Hamiltonian with $\vec{\lambda} = \left[-\frac{\partial V}{\partial x_1}, -\frac{\partial V}{\partial x_2}, \cdots, -\frac{\partial V}{\partial x_n}\right]$.
3. Reconstruct the optimal trajectory from the provided initial conditions.

Viscosity Solutions

As mentioned earlier, the Hamilton-Jacobi-Bellman equation admits only C^1 smooth solutions, which limits the applicability of the technique severely. Hence, the notion of viscosity solutions was introduced by Crandall and Lions in the early 1980s [49, 50]. In the viscosity solution framework, the standard derivatives are replaced by super- and subdifferentials, and the actual solution is considered to lie between the two solutions generated by the super- and subdifferentials. By finding the solution in this manner, the smoothness assumptions can be relaxed.

Definition: Viscosity Super- and Subsoltions:

A function $v \in C([t_0, T] \times \mathbb{R}^n)$ is called a viscosity supersolution of equation 40 if

$$v(T, x) \geq \varphi(\vec{x}(T)), \qquad \forall x \in \mathbb{R}^n,$$

and for any $v' \in C^1([t_0, T] \times \mathbb{R}^n)$, whenever v - v' attains a local minimum at $(t, \vec{x}) \in [t_0, T] \times \mathbb{R}^n$, we have

$$\frac{\partial v'}{\partial t} + sup_{\vec{u} \in U} \left\{ \langle - \left[\frac{\partial v'}{\partial x_1}, \frac{\partial v'}{\partial x_2}, \cdots, \frac{\partial v'}{\partial x_n} \right], f(\vec{x}, \vec{u}, t) \rangle + L(\vec{x}, \vec{u}, t) \right\} \geq 0$$

Similarly, a function $v \in C([t_0, T] \times \mathbb{R}^n)$ is called a viscosity subsolution of Equation 42 if

$$v(T, x) \leq \varphi(\vec{x}(T)), \qquad \forall x \in \mathbb{R}^n,$$

and for any $v' \in C^1([t_0, T] \times \mathbb{R}^n)$, whenever v - v' attains a local maximum at $(t, \vec{x}) \in [t_0, T] \times \mathbb{R}^n$, we have

$$\frac{\partial v'}{\partial t} + sup_{\vec{u} \in U} \left\{ \langle - \left[\frac{\partial v'}{\partial x_1}, \frac{\partial v'}{\partial x_2}, \cdots, \frac{\partial v'}{\partial x_n} \right], f(\vec{x}, \vec{u}, t) \rangle + L(\vec{x}, \vec{u}, t) \right\} \leq 0.$$

Finally, in order to relate the super- and subsolutions of Equation 42 to the value function, it is important to define the super- and subdifferentials.

Definition: Super- and Subdifferentials:

The superdifferential is given by:

$$D^{1,+}_{t,\vec{x}} v(t,\vec{x}) =$$

$$\left\{ \left(dt_s, \vec{\lambda}_s\right) \in \mathbb{R} \times \mathbb{R}^n \middle| \lim_{\substack{s \to t, s \in [t_0, T) \\ \vec{y} \to \vec{x}}} \sup \left(\frac{v(s,\vec{y}) - v(t,\vec{x}) - dt_s \cdot (s-t) - \langle \vec{\lambda}_s, \vec{y} - \vec{x} \rangle}{|s-t| + |\vec{y} - \vec{x}|} \right) \leq 0 \right\}$$

and the subdifferential is given by:

$$D^{1,-}_{t,\vec{x}} v(t,\vec{x}) =$$

$$\left\{ \left(dt_s, \vec{\lambda}_s\right) \in \mathbb{R} \times \mathbb{R}^n \middle| \lim_{\substack{s \to t, s \in [t_0, T) \\ \vec{y} \to \vec{x}}} \inf \left(\frac{v(s,\vec{y}) - v(t,\vec{x}) - dt_s \cdot (s-t) - \langle \vec{\lambda}_s, \vec{y} - \vec{x} \rangle}{|s-t| + |\vec{y} - \vec{x}|} \right) \geq 0 \right\}$$

The super- and subsolutions and the super- and subdifferentials notions can be related to the value function by the following theorem.

Theorem 2:

The value function $V \epsilon\, C([t_0, T] \times \mathbb{R}^n)$ is the only function that $\forall (t, \vec{x}) \in [t_0, T] \times \mathbb{R}^n$ satisfy the following:

$$\begin{cases} dt_s + \sup_{\vec{u} \in U} H(t, \vec{x}, \vec{u}, -\vec{\lambda}_s) \leq 0, \forall \left(dt_s, \vec{\lambda}_s\right) \in D^{1,+}_{t,\vec{x}} v(t,\vec{x}) \\ dt_s + \sup_{\vec{u} \in U} H(t, \vec{x}, \vec{u}, -\vec{\lambda}_s) \geq 0, \forall \left(dt_s, \vec{\lambda}_s\right) \in D^{1,-}_{t,\vec{x}} v(t,\vec{x}) \\ V\left(T, \vec{x}(T)\right) = \varphi\left(\vec{x}(T)\right) \end{cases}$$

Since theorem 2 is derived from Bellman's optimality principle, the two assumptions made when Bellman's optimality principle was derived must also hold for theorem 2.

Maximum Principle Approach

Instead of determining the optimal control strategy for an entire family of problems, the Maximum Principle utilizes information about the starting and termination conditions along with the Hamiltonian to determine a particular solution to the optimal control problem. In essence, the Maximum Principle constructs only a single optimal trajectory instead of deriving the value of the problem over the entire state space.

The maximum principle relies on the zero-derivative condition for the unconstrained optimization problem and the Karush-Kuhn-Tucker condition for the non-linear constrained optimization condition to establish a set of necessary conditions for an optimal solution. As mentioned in the dynamic programming subsection, the Hamiltonian is given by:

$$H\left(t,\vec{x}(t),\vec{u}(t),\vec{\lambda}(t)\right) = \langle\vec{\lambda}(t), f(\vec{x}(t),\vec{u}(t),t)\rangle + L(\vec{x}(t),\vec{u}(t),t).$$

The maximum principle for a non-stochastic optimization problem described by Equation 39 and Equation 40, is given by the following set of differential equations:

$$\dot{\vec{x}}(t) = f(\vec{x}(t),\vec{u}(t),t) = \frac{\partial H\left(t,\vec{x}(t),\vec{u}(t),\vec{\lambda}(t)\right)}{\partial\vec{\lambda}(t)}$$

$$\dot{\vec{\lambda}}(t) = -\frac{\partial H\left(t,\vec{x}(t),\vec{u}(t),\vec{\lambda}(t)\right)}{\partial\vec{x}(t)} = -\left(\frac{\partial f(\vec{x}(t),\vec{u}(t),t)}{\partial\vec{x}(t)}\right)^T \cdot \vec{\lambda}(t) - \frac{\partial L(\vec{x}(t),\vec{u}(t),t)}{\partial\vec{x}(t)}$$

$$x(t_0) = x_0, \vec{\lambda}(T) = \frac{\partial\varphi\left(\vec{x}(T)\right)}{\partial\vec{x}(T)}$$

$$H\left(t,\vec{x}(t),\vec{u}(t),\vec{\lambda}(t)\right) = \max_{\vec{u}'\in U} H\left(t,\vec{x}(t),\vec{u}',\vec{\lambda}(t)\right)$$

$$\frac{\partial H\left(t,\vec{x}(t),\vec{u}(t),\vec{\lambda}(t)\right)}{\partial\vec{u}(t)} = \left(\frac{\partial f(\vec{x}(t),\vec{u}(t),t)}{\partial\vec{u}(t)}\right)^T \vec{\lambda}(t) + \frac{\partial L(\vec{x}(t),\vec{u}(t),t)}{\partial\vec{u}(t)} = 0$$

where U is the set of admissible controls for which $\vec{u}$ is measureable. It should be noted that the number of boundary conditions must equal the number of differential equations, that is, if *n* differential equations must be solved, the $x(t_0)$ and $\vec{\lambda}(T)$ constraints must provide at least *n* boundary conditions.

In order to derive the above system of equations, the following three assumptions must hold:

1. (U, d) is a separable metric space.
2. *f(), L(),* and $\varphi()$ are measureable, and there $\exists k \in \mathbb{R}$, $k > 0$ and a modulus of continuity $\bar{\omega}: [0,\infty) \rightarrow [0,\infty)$ such that for $\alpha(\vec{x},\vec{u},t) = f(\vec{x},\vec{u},t), \alpha(\vec{x},\vec{u},t) = L(\vec{x},\vec{u},t)$ or $\alpha(\vec{x},\vec{u},t) = \varphi(\vec{x})$,
$$\begin{cases} \left|\alpha(\vec{x},\vec{u},t) - \alpha\left(\hat{\vec{x}},\hat{\vec{u}},t\right)\right| \leq k \cdot \left|\vec{x} - \hat{\vec{x}}\right| + \bar{\omega}\left(d\left(\hat{\vec{u}},\vec{u}\right)\right), \forall t \in [t_0,T], \vec{x},\hat{\vec{x}} \in \mathbb{R}^n, \vec{u},\hat{\vec{u}} \in U \\ \qquad |\alpha(0,\vec{u},t)| \leq k, \forall (t,\vec{u}) \in [t_0,T] \times U. \end{cases}$$
3. *f(), L(),* and $\varphi()$ are C^1 continuous in *x*, and there exist a modulus of continuity $\bar{\omega}: [0,\infty) \rightarrow [0,\infty)$ such that for $\alpha(\vec{x},\vec{u},t) = f(\vec{x},\vec{u},t), \alpha(\vec{x},\vec{u},t) = L(\vec{x},\vec{u},t)$ or $\alpha(\vec{x},\vec{u},t) = \varphi(\vec{x})$,
$$\left|\frac{\partial\alpha(\vec{x},\vec{u},t)}{\partial\vec{x}} - \frac{\partial\alpha\left(\hat{\vec{x}},\hat{\vec{u}},t\right)}{\partial\vec{x}}\right| \leq \bar{\omega}\left(\left|\vec{x} - \hat{\vec{x}}\right| + d\left(\hat{\vec{u}},\vec{u}\right)\right),$$
$$\forall t \in [t_0,T], \vec{x},\hat{\vec{x}} \in \mathbb{R}^n, \vec{u},\hat{\vec{u}} \in U$$

The maximum principle is a very powerful tool to solve optimal control problems; however, since some of the boundary conditions are given as initial conditions and others at termination, solving the system of differential equations is not easy. In addition, the differentiability assumption may not be a valid assumption for a particular problem.

The mixed initial and final boundary constraints problem, can be solved using the shooting method [40, 41]. The shooting method approach attempts to iteratively convert the given two-point boundary value problem into an initial value problem, by successively guess initial values and, based on the resulting solution, improve the guess until a suitable solution is reached. Another approach to solving the two-point boundary value problem, is to rely on Adomian decomposition as described in [39].

Minimum Time Problems

Minimum time problems are a special case of the optimization problem described by equation 38 and 39. The objective is to reach a given goal as fast as possible, that is, the termination time is not given but is instead a variable that must be minimized. A typical minimum time cost function is given in Equation 41.

One of the properties of the minimum-time problem that distinguishes it from other optimal control problems is the termination condition. In a fixed time problem, time is used as a termination condition; however, since no final time is given in the minimum-time problem, a different termination condition *must* be provided:

$$\Psi(\vec{x}(T), T) = 0$$

Minimum-Time Maximum Principle Formulation

In a minimum-time optimal control problem, L = 1 and φ = 0, which implies that the Maximum principle formulation can be changed to:

$$\begin{gathered}
\dot{\vec{x}}(t) = f(\vec{x}(t), \vec{u}(t), t) = \frac{\partial H\left(t, \vec{x}(t), \vec{u}(t), \vec{\lambda}(t)\right)}{\partial \vec{\lambda}(t)} \\
\dot{\vec{\lambda}}(t) = -\frac{\partial H\left(t, \vec{x}(t), \vec{u}(t), \vec{\lambda}(t)\right)}{\partial \vec{x}(t)} = -\left(\frac{\partial f(\vec{x}(t), \vec{u}(t), t)}{\partial \vec{x}(t)}\right)^T \cdot \vec{\lambda}(t) \\
x(t_0) = x_0, \left(\vec{\lambda}(T)\right)^T \cdot f(\vec{x}(T), \vec{u}(T), T) = -1 \\
H\left(t, \vec{x}(t), \vec{u}(t), \vec{\lambda}(t)\right) = \max_{\vec{u}' \in U} H\left(t, \vec{x}(t), \vec{u}', \vec{\lambda}(t)\right) \\
\frac{\partial H\left(t, \vec{x}(t), \vec{u}(t), \vec{\lambda}(t)\right)}{\partial \vec{u}(t)} = \left(\frac{\partial f(\vec{x}(t), \vec{u}(t), t)}{\partial \vec{u}(t)}\right)^T \vec{\lambda}(t) = 0
\end{gathered} \tag{43}$$

Notice that the termination condition has changed due to the change in the termination condition. When time was used as a termination condition, the following expression was used to determine the value of the costate $\vec{\lambda}(t)$ at termination:

$$\vec{\lambda}(T) = \frac{\partial \varphi\left(\vec{x}(T)\right)}{\partial \vec{x}(T)}$$

However, since we have introduced a constraint at termination, the new expression becomes:

$$\vec{\lambda}(T) = \frac{\partial \varphi\left(\vec{x}(T)\right)}{\partial \vec{x}(T)} + \gamma \cdot \frac{\partial \Psi(\vec{x}(T), T)}{\partial \vec{x}(T)}$$

However the additional condition introduced into the problem adds a term into the cost variation and by ensuring that this term is zero we obtain the expression,

$$\frac{\partial\varphi\big(\vec{x}(T)\big)}{\partial t} + \gamma\cdot\frac{\partial\Psi(\vec{x}(T),T)}{\partial t} + L(\vec{x}(T),\vec{u}(T),T) + \left(\vec{\lambda}(T)\right)^T \cdot f(\vec{x}(T),\vec{u}(T),T) = 0 \Leftrightarrow$$
$$1 + \left(\vec{\lambda}(T)\right)^T \cdot f(\vec{x}(T),\vec{u}(T),T) = 0 \Leftrightarrow$$
$$\left(\vec{\lambda}(T)\right)^T \cdot f(\vec{x}(T),\vec{u}(T),T) = -1$$

as given in equation (42).

<u>Minimum-Time Hamilton-Jacobi-Bellman Formulation</u>

The Hamilton-Jacobi-Bellman equation does not change significantly in the minimum-time problem. The system to be solved is expressed by the following system,

$$\begin{cases} 1 + sup_{\vec{u}\in U}\{\langle\vec{\lambda}, f(\vec{x},\vec{u},t)\rangle\} = 0 \\ v|_{t=T} = \varphi\big(\vec{x}(T)\big) \end{cases}, \big(t,\vec{x},\vec{u},\vec{\lambda}\big) \in [t_0,T]\times\mathbb{R}^n\times U\times\mathbb{R}^n$$

The termination condition, $\Psi(\vec{x}(T),T) = 0$, provides the boundary conditions needed to back-propagate the solution in time. In essence, the termination condition can be viewed as an interface that is being propagated backwards in time until it covers a predetermined part of the state space.

REFERENCES

[1] CHIZEK, J. G. “Military Transformation: Intelligence, Surveillance and Reconnaissance,” Congressional Research Service, 2003.

[2] CUMMINGS, M., NEHME, C., and CRANDALL, J., “Predicting Operator Capacity for Supervisory Control of Multiple UAVs,” Aerospace Science and Technology, http://web.mit.edu/aeroastro/www/labs/halab/papers/CummingsNehmeCrandall_2006.pdf (Accessed: 04/18/07)

[3] OFFICE OF THE SECRETARY OF DEFENSE, “OSD UAV Roadmap 2005-2030,” Aug. 2005.

[4] STIPANOVIC, D., SRIRAM, and TOMLIN, C. “Strategies for Agents in Multi-Player Pursuit-Evasion Games,” Symposium on Dynamic Games and Applications, Arizona, 2004.

[5] GUTIN, G., and PUNNEN, A., “The Traveling Salesman Problem and Its Variations,” Springer, 2006.

[6] ARORA, “Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and other Geometric Problems,’ Journal of ACM, 45, pp. 753-782, 1998.

[7] OLFATI-SABER, R., “Flocking for Multi-Agent Dynamic Systems: Algorithms and Theory,” IEEE Transactions on Automatic Control, Vol. 51, No. 3, March 2006.

[8] LI, X., XIAO, J., AND CAI, Z., “Stable Flocking of Swarms Using Local Information,” IEEE International Conference on Systems, Man & Cybernetics, p. 3921 – 3927, 2005.

[9] GERKEY, B., and MATARIC, M., “A framework for studying multi-robot task allocation’, Proceedings of the NRL Workshop on Multi-robot Systems, 2003.

[10] BRUCE, J., BOWLING, M., BROWNING, B., and VELOSO, M., “ Multi-Robot Team Response to a Multi-Robot Opponent Team,” Proceedings of the 2003 IEEE International Conference on Robotics and Automation, Taipei, Taiwan, Sept. 2003.

[11] PAGELLO, E., D'ANGELO, A., and MENEGATTI, E., "Cooperation Issues and Distributed Sensing for Multirobot Systems," Proceedings of the IEEE, Vol. 94, No. 7, July 2006.

[12] BENJAMIN, M. R., "Multi-objective Autonomous Vehicle Navigation in the Presence of Cooperative and Adversarial Moving Contacts," OCEANS MTS/IEEE, 2002.

[13] AXELSSON, H., EGERSTEDT, M., and WARDI, Y., "Reactive Robot Navigation Using Optimal Timing Control," American Control Conference, Portland, USA, 2005.

[14] PAULUMBO, N., REARDON, B., and BLAUWKAMP, R., "Integrated Guidance and Control for Homing Missiles," John Hopkins APL Technical Digest, Vol. 25, No. 2, 2004.

[15] EKLUND, J., SPRINKLE, J., and SASTRY, S., "Implementing and Testing a Nonlinear Model Predictive Tracking Controller for Aerial Pursuit/Evasion Games on a Fixed Wing Aircraft," American Control Conference, June 2005.

[16] HESPANHA, J., PRANDINI, M., and SASTRY, S., "Probabilistic Pursuit-Evasion Games: A One-Step Nash Approach," Proceedings of the 39th IEEE Conference on Decision and Control, Australia, Dec. 2000.

[17] ISAACS, R., *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit*. New York: John Wiley and Sons, Inc, 1965

[18] BARDI, M., FALCONE, M., and SORVIA, P., " Numerical Methods for Pursuit-Evasion Games via Viscosity Solutions," Annals of the International Society of Dynamic Games, Birkhäuser, 1999.

[19] LACHNER, R., BREITNER, M., and PESCH, H., "Three-Dimensional Air Combat: Numerical Solution of Complex Differential Games," Annals of the International Society of Dynamic Games, Birkhäuser, 1995.

[20] MITCHELL, I., BAYEN., A, and TOMLIN, C., "A Time-Dependent Hamilton-Jacobi Formulation of Reachable Sets for Continuous Dynamic Games," IEEE Transactions on Automatic Control, Vol. 50, No. 7, July 2005.

[21] TEO, R., JANG, J., and TOMLIN, C., "Automated Multiple UAV Flight – the Stanford DragonFly UAV Program," 43rd IEEE Conference on Decision and Control, Dec. 2004.

[22] KAO, C., OSHER, S., and QIAN, J., "Lax-Friedrich sweeping scheme for static Hamilton-Jacobi equations," Journal of Computational Physics, Vol. 196, issue 1, pp. 367-391, 2004.

[23] VIDAL, R., SHAKERNIA, O., KIM, J., SHIM, D., AND SASTRY, S., "Probabilistic Pursuit-Evasion Games: Theory, Implementation, and Experimental Evaluation," IEEE Transactions on Robotics and Automation, Vol. 18, No. 5, Oct. 2002.

[24] YAMAUCHI, B., "Decentralized coordination for multirobot exploration," Robotics and Autonomous Systems, pp. 111-118, 1999.

[25] ISLER, V., KANNAN, S., and KHANNA, S., "Randomized Pursuit-Evasion in a Polygonal Environment," IEEE Transactions on Robotics, Vol. 21, No. 5, Oct 2005.

[26] ISLER, V., KANNAN, S., and KHANNA, S., "Locating and Capturing an Evader in a Polygonal Environment," Technical Reports (CIS), http://repository.upenn.edu/cis_reports/18, University of Pennsylvania, 2005.

[27] SUBBOTIN, A., "Constructive Theory of Positional Differential Games and Generalized Solutions to Hamilton-Jacobi Equations," Annals of the International Society of Dynamic Games, Birkhäuser, 1999.

[28] MELIKYAN, A. *Generalized Characteristics of First Order PDEs: Applications in Optimal Control and Differential Games.* Birkhäuser Boston, 1998.

[29] POZNYAK, A., and GALLEGOS, C., "Multimodel Prey-Predator LQ Differential Games," Proceedings of the American Control Conference, June 2003.

[30] SOUGANIDIS, P., "Zero-sum differential games and viscosity solutions," Annals of the International Society of Dynamic Games, Birkhäuser, 1999.

[31] L. C. EVANS and P. E. SOUGANIDIS, "Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations," Indiana University of Mathematics Journal, p. 773-797, 1984.

[32] SETHIAN, J., "Adaptive Fast Marching and Level Set Methods for Propagating Interfaces," Acta Math. Univ. Comenianae Vol LXVII, pp. 3-15, 1998.

[33] SETHIAN, J., and VLADIMIRSKY, A., "Ordered Upwind Method for Static Hamilton-Jacobi Equations: Theory and Algorithms," SIAM Journal of Numerical Analysis, Vol. 41, No. 1, pp. 325-363, 2003.

[34] GIGA, Y. *Surface evolution equations: A level set approach.* Birkhäuser Boston, 2006.

[35] BRYSON, A., and HO, Y. *Applied Optimal Control: Optimization, Estimation, and Control*. Hemisphere Publishing Corporation, 1975.

[36] GALLEGOS, C., "Solving Coupled Riccati Equation for the N-Player LQ Differential Game," 2nd International Conference on Electrical and Electronics Engineering, 2005.

[37] TURETSKY, V., and SHINAR, J., "Missile guidance laws based on pursuit-evasion game formulations," Automatica 39, pp. 607-618, 2003.

[38] ENGWERDA, J., "On the open-loop equilibrium in LQ-games", Journal of Economic Dynamics and Control, pp. 729-726, 1998.

[39] KUMAR, P., AND SCHUPPEN, J., "On Nash Equilibrium Solutions in Stochastic Dynamic Games", IEEE Transactions on Automatic Control, Vol. ac-24, No. 6, Dec. 1980.

[40] MAITRA, A., and SUDDERTH, W., "An introduction to Gambling Theory and Its Applications to Stochastic Games," Annals of the International Society of Dynamic Games, Birkhäuser, 1999.

[41] BOLTYANSKY, V., and POZNYAK, A., "Robust maximum principle in minimax control,' International Journal of Control, Vol. 72, No. 4, pp. 305-314, 1999.

[42] EKLUND, J., SPRINKLE, J., and SASTRY, S., "Implementing and Testing a Nonlinear Model Predictive Tracking Controller for Aerial Pursuit/Evasion Games on a Fixed Wing Aircraft," American Control Conference, 2005.

[43] WAZWAZ, A., "A Reliable Algorithm for Obtaining Positive Solutions for Nonlinear Boundary Value Problems," Computers and Mathematics with Applications 41, pp 1237-1244, 2001.

[44] TRENT, A., VENKATARAMAN, R., and DOMAN, D., "Trajectory Generation Using A Modified Simple Shooting Method," IEEE Aerospace Conference Proceedings, 2004.

[45] HOLSAPPLE, R., and VENKATARAMAN, R., "New, Fast Numerical Method for Solving Two-Point Boundary Value Problems," Journal of Guidance, Vol. 27, No. 2, pp. 301-304, 2004.

[46] LAW, M., and CERRATO, M., "Improving Local Refinement Algorithms for Adaptive Meshing of Process Simulation Problems," Simulation of Semiconductor Processes and Devices, 1997.

[47] DEYING, X., and LIU, X., "Energy Efficient Broadcast Routing in Static Ad Hoc Wireless Networks," IEEE Transactions on Mobile Computing, Vol. 3, No. 2, 2004.

[48] WIESELTHIER, J., NGUYEN, G., and EPHREMIDES, A., "The Energy Efficiency of Distributed Algorithms for Broadcasting in Ad Hoc Networks," The 5th International Symposium on Wireless Personal Multimedia Communications, 2002.

[49] BARDI, M., and CAPPUZZO-DOLCETTA, I. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations.* Birkhäuser Boston, 1997.

[50] YONG, J., and ZHOU, X, *Stochastic Controls: Hamiltonian Systems and HJB Equations.* Springer-Verlag New York, 1999.

Related Publications

Ludington, B., Reimann, J., and Vachtsevanos, G., "Target Tracking and Adversarial Reasoning for Unmanned Aerial Vehicles", IEEE Aerospace Conference, Big Sky, Montana, 2007.

Reimann, J., Vachtsevanos, G., Ge, J., and Tang, L., "An Approach to controlling swarms of unmanned aerial vehicles in adversarial situations". Paper presented at the AIAA Guidance, Navigation and Control Conference and Exhibit, Colorado, 2006.

Ge, J., Tang, L., Reimann, J., and Vachtsevanos, G., "Suboptimal approaches to multiplayer pursuit-evasion differential games": Paper presented at the AIAA Guidance, Navigation and Control Conference and Exhibit, Colorado, 2006.

Ludington, B., Reimann, J., Barlas, I., and Vachtsevanos, G., "Target tracking with unmanned aerial vehicles: From single to swarm vehicle autonomy and intelligence", Proceedings of the 14th Mediterranean Conference on Control and Automation, Ancona, Italy, 2006.

Reimann, J. and Vachtsevanos, G., "UAVs in urban operations: Target Interception and Containment". Journal of Intelligent and Robotic Systems, Springer, p. 383-396, 2006.