Path-Planning Strategies for Ambush Avoidance

by

Farmey A. Joseph

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2005

Author .....
Department of Aeronautics and Astronautics
July 22, 2005

Certified by .....
Eric Feron
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Jaime Peraire
Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Studies

Path-Planning Strategies for Ambush Avoidance

by

Farmey A. Joseph

Submitted to the Department of Aeronautics and Astronautics
on July 22, 2005, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

This thesis examines a variety of ambush games in which one player must navigate between an origin and a destination, and the other player seeks to intercept and ambush him. These games include single-stage games, in which all decisions are made at the outset of the game, and multi-stage games, in which the second player may choose his ambush locations based on real-time updates of the first player's position. For both types of ambush games, methods are presented for efficiently computing the optimal mixed strategies for the first player to navigate between the origin and destination, so as to minimize the probability of being ambushed. The results are applicable to a wide range of real-life situations, including the routing of VIPs and convoys through hostile areas.

Thesis Supervisor: Eric Feron
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

I must begin by thanking my advisor, Eric Feron, for all he has given me over the last two years. I made the decision to begin graduate school at MIT largely for the opportunity of working with him, and now that I've come to the end, I am happy to say that I am not disappointed with my decision. Eric most generously allowed me the freedom to pursue my academic interests, and this thesis is a direct result of that.

Certainly, the biggest fringe benefit of studying under Eric has been the friendship and mentorship provided by his other students. Jan De Mot and Tom Schouvenaars were the ideal PhD candidates in my eyes, and if I ever choose to become a PhD student myself, it will be largely due to their inspiration. Vlad and Rodin patiently imparted to me some of their huge store of practical engineering knowledge. Masha and Phil gave me some key advice for my research. The rest of my lab mates, including Emily, Glenn, Greg, Han-Lim, Ji Hyun, Mardavij, Mario, Navid, Olivier, Selcuk, and others all contributed to my time here, and I am very happy to have known them.

I must express my gratitude to the faculty at MIT. I was very fortunate to work with John Deyst and Karen Willcox as a teaching assistant, and I will always remember that experience as one of my most enjoyable at MIT. Asuman Ozdaglar gave me some key insights into game theory and taught me everything I needed to know about linear optimization.

I also must thank my parents, who are ultimately responsible for my achievements. It was a difficult decision for them to send me to the other side of the world to pursue my education, and I hope they are satisfied with the results! I am grateful also to all my family and friends, both in the US and in Australia, and particularly to Lily, who suffered through undergrad with me and remains the finest friend anyone can ask for.

Finally, although IHTFP as much as anyone else, it would be remiss of me not to thank MIT and its donors for giving me the opportunity to come to the United States and receive a world-class engineering education. I will return to Australia with very fond memories of my time here.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

It is an unfortunate fact of today's world that there are many situations in which a person must travel from one point to another while running the risk of ambush by hostile forces. That person is especially vulnerable if he must make the journey between the same two locations on a regular schedule. An obvious example is a VIP who must travel from his residence to his place of work at roughly the same time every morning, and then return that evening. To ensure his own survival, the VIP must have a sufficiently strong armed escort to deter or defeat any ambush attempt, or he must evade the enemy by choosing a route that does not pass through the place of ambush.

Similar examples that occur in modern conflict zones (e.g. Iraq) include: Military Police who must drive from their base to a designated location each day to provide security, a humanitarian convoy that must deliver aid every day from a supply base to a designated aid dispersal site, ambassadors who work at the embassy but live elsewhere, etc.

An important feature of these scenarios is that the relatively small-scale of the scenario allows little or no opportunity for deception. Most VIPs, MP units, convoys, etc. do not command the resources for decoy vehicles, nor is it always practical or useful to use their own vehicles to "feint" along one route but ultimately choose a different route. The hostile forces, for their part, cannot place fake ambushes, and even if they did, it is unlikely that their opponents would have the means to detect

them.

Therefore, the agent making the journey is faced with a problem of pure randomization. Specifically, how to randomize his routing to make it as difficult as possible for his opponent to ambush him? This problem of determining the optimal randomization is far from trivial, particularly when there are literally thousands of possible routes between the origin and the destination.

To the author's knowledge, it appears that most security staff responsible for VIP security rely on their intuition and experience to select a different route each day.[1] While that is superior to not randomizing at all, and on some occasions may be more than adequate, it would be desirable to have some tool available to rigorously determine the optimal randomized routing strategy for the given scenario. Of course, nothing could ever completely guarantee the safety of the VIP, because no matter how well he randomizes his routing the probability of his being ambushed will always be greater than zero. However, it is still worthwhile to minimize this probability, and give the VIP the best possible chance of survival.

The major contribution of this thesis is to develop such a tool for a variety of different ambush games. Chapter 2 includes a review of previous work in the field of determining optimal strategies for ambush games. Chapter 3 examines *single-stage* ambush games in which the decision on which route to take and the decision on ambush placement are made at the outset of the game. Chapter 4 examines *multi-stage* ambush games in which the enemy can observe the agent's position and act on that information as he travels from origin to destination. For both types of ambush games, methods are presented to efficiently compute the optimal routing strategy. Finally, Chapter 5 concludes this thesis with a summary of results and suggestions for future work.

Although the examples in this thesis are concerned with ground transportation, both air and sea transport may also be vulnerable to ambush. The threat of surface-to-air missiles (SAMs) against commercial aviation is a real one, as shown by the attack on an airliner in Kenya in 2002. Since aircraft are most vulnerable to SAMs

[1] As an example, see http://www.securitydriver.com/aic/index.html

when flying close to the ground, it may be advantageous to randomize the arrival and departure routing at certain airports. Meanwhile, piracy continues to present a serious problem for the shipping industry in several parts of the world, such as the Malacca Strait. Randomized routing of cargo ships could reduce the probability of being intercepted and boarded by armed pirates. The methods presented in this thesis can be adapted to these scenarios, and indeed any scenario involving the risk of ambush where the sample area can be discretized.

Chapter 2

Previous Work

Game theory has been applied to problems of ambush for many years [13]. A typical ambush game, such as the VIP transport problem referred to in Chapter 1, can be thought of as a non-cooperative, zero-sum, two-player game. Player 1 (the VIP) must choose a route from origin to destination. Player 2 (the hostile forces) must choose some number of locations at which to prepare ambushes. If Player 1's path passes through an ambush site, then Player 2 wins. If Player 1's path avoids all ambush sites, then Player 1 wins.

One option for Player 1 is to attempt to deceive Player 2 by manipulating the information available to him. For example, Player 1 could launch a decoy vehicle, or alternatively use his own vehicle to "feint" and then backtrack. In [7], the authors demonstrate that deception can be used in such games to increase the expected pay-off for one of the players, although, somewhat counter-intuitively, the effectiveness of deception decreases when one player has too much control over the information available to the other.

An example of the use of deception in a specific military scenario is given in [9] and [10]. A convoy must journey from one point to another, and small UAVs are used to overfly the convoy route together with several deception routes in such a way as to obtain reconnaissance information while simultaneously deceiving the enemy as to the intended convoy route.

Although deception can be a valuable tool for a person who seeks to avoid ambush,

it is not always a feasible option, as discussed in Chapter 1. Instead, the assumption throughout this thesis is that neither player can manipulate the information available to the other.

It is unlikely that Player 1's optimal course of action is to choose the same route every time the game is played, nor is it likely that Player 2's optimal course of action is to choose the same ambush locations every time. For this type of game, the optimal strategy for each player tends to be a mixed strategy rather than a pure strategy. Therefore, Player 1's optimal strategy will consist of a set of probabilities for choosing among all possible routes from origin to destination. Player 2's optimal strategy will consist of a set of probabilities for choosing among all possible ambush sites.

The most straightforward way to solve for the optimal mixed strategies is to construct the corresponding game matrix $\mathbf{A}$. The rows of $\mathbf{A}$ correspond to all possible strategies for Player 1, which are the accessible routes from origin to destination. The columns of $\mathbf{A}$ correspond to all possible strategies for Player 2, which are feasible ambush locations. A_{ij} is then equal to the game outcome if Player 1 pursues strategy i and Player 2 pursues strategy j. If Player 1's mixed strategy is $\mathbf{p}$ and Player 2's mixed strategy is $\mathbf{q}$, then the expected game outcome is $\mathbf{p'Aq}$. Once the game matrix is available, the optimal strategies $\mathbf{p}^*$ and $\mathbf{q}^*$ can be solved for using a linear program, as described in [1] or [8]:

$$\begin{aligned} \text{maximize} \quad & \mathbf{1}'\tilde{\mathbf{p}} \\ \text{subject to} \quad & \mathbf{A}'\tilde{\mathbf{p}} \leq \mathbf{1} \\ & \tilde{\mathbf{p}} \geq 0 \end{aligned} \tag{2.1}$$

$$\Longrightarrow \quad \mathbf{p}^* = \frac{\tilde{\mathbf{p}}^*}{\mathbf{1}'\tilde{\mathbf{p}}^*}$$

$$\begin{aligned} \text{maximize} \quad & \mathbf{1}'\tilde{\mathbf{q}} \\ \text{subject to} \quad & \mathbf{A}'\tilde{\mathbf{q}} \leq \mathbf{1} \\ & \tilde{\mathbf{q}} \geq 0 \end{aligned} \tag{2.2}$$

$$\Longrightarrow \quad \mathbf{q}^* = \frac{\tilde{\mathbf{q}}^*}{\mathbf{1}'\tilde{\mathbf{q}}^*}$$

where $\tilde{\mathbf{p}}^*$ and $\tilde{\mathbf{q}}^*$ are the solutions to the two LPs, and $\mathbf{p}^*$ and $\mathbf{q}^*$ are the optimal strategies for the two players.

Although this approach is conceptually simple, there are two severe limitations that prevent it from being applied to many real-world ambush problems. The first is that it requires a list to be made of all possible routes from origin to destination, which can be a massive problem in itself for a large road network. The second is that the size of the matrix $\mathbf{A}$ and the number of decision variables grow exponentially with the size and complexity of the scenario map. For a real-life ambush scenario that takes place in a city with a dense road network, there may be many thousands of possible routes from origin to destination, and a large number of possible ambush locations as well. The resulting LP requires massive storage and computational effort to solve.

As an example, this approach was applied to a scenario involving a map of Cambridge in order to demonstrate its inefficiency. With 50 street intersections, approximately 450 different routes existed between the origin and destination of the scenario. When Player 2 could choose 3 ambush sites, the resulting game matrix had a size of 450x19600. The associated LP overwhelmed the memory resources of a desktop PC and could not be solved.

A number of researchers have avoided these limitations by finding exact solutions for various ambush games. In [13], the authors consider an ambush game that takes place on a rectangular lattice, in which one player must choose a path from one side of the rectangle to the other, and his opponent places barriers of finite length along

certain columns inside this rectangle. Optimal strategies for each player are found for a number of different cases that specify the freedom of movement of the first player and the number of barriers available to the second player. A continuous version of this game is discussed in [11] and [14], and further cases of this game are solved in [2]. Problem formulations and solutions for these and other similar games can be found in [12].

[4] and [16] solve discretized versions of one of these games in which the second player sets two barriers, and these results are extended for the n-barrier case in [15].

Although these results are significant and cover a wide range of ambush games, it is difficult to apply them to an arbitrary graph that represents the layout of a non-uniform city road network.

Chapter 3

The Single-Stage Ambush Game

In this section, we find a computationally efficient solution to a single-stage ambush game that may model the VIP transport problem. This game can be thought of as a zero-sum game between two players. Player 1 (the VIP) must choose a route from origin to destination. Player 2 (the hostile forces) must choose one or more locations, depending on his resources, at which to prepare an ambush. If Player 1's path passes through an ambush site, then Player 2 wins, and the outcome of the game is dependent on the vulnerability of that location to an ambush. If Player 1's path avoids all ambush sites, then Player 1 wins, and the outcome of the game is zero. This game is classified as a single-stage game because both players make their decisions in one stage at the outset of the game.

The objective is to determine the optimal routing strategy for Player 1 so as to minimize the expected outcome of the game. Intuitively, it is highly unlikely that his optimal strategy will be a pure strategy (i.e. choosing a single course of action with probability 1) because that would imply choosing the same route from origin to destination on every occasion, which would allow Player 2 to achieve a 100% success rate of ambushing Player 1. Instead, Player 1's optimal strategy is likely to be a *mixed strategy*, which assigns a probability to each possible route. Every time the game is played, Player 1 would then choose a route at random based on the probabilities given by this mixed strategy.

The terms "optimal strategy" and "expected outcome" need some clarification

here, because obviously they are also dependent on Player 2's chosen strategy. Indeed, Player 1's optimal strategy is a function of Player 2's chosen strategy, and vice versa, and the expected outcome of the game varies accordingly.

In a game between two intelligent players, the rational course of action for one player is therefore to choose the strategy that will minimize his losses across all of the other player's strategies. In this case, in forming his optimal routing strategy, Player 1 should assume that Player 2 will place his ambushes in the worst possible location from Player 1's point of view. This solution is referred to as a *minimax solution* because Player 1 is minimizing the maximum outcome of the game, by choosing the strategy whose worst outcome is the minimum worst outcome across all strategies.

From Player 2's point of view, his own optimal strategy is the *maximin solution*. That is, for placing his ambushes, he should choose the strategy whose worst outcome is the maximum worst outcome across all strategies.

By the Minimax Theorem (e.g. see [1]), the expected outcomes of the minimax and maximin solutions are equal. If Player 1 pursues his minimax strategy and Player 2 his maximin strategy, the result is a *saddle-point equilibrium* because each strategy is optimal against the other, and neither player can unilaterally change his strategy without making himself worse off.

The minimax strategy is therefore the "optimal strategy" that we seek to find in the following ambush games. The "expected outcome" of the game refers to the minimax value or (equivalently) the expected outcome at the saddle-point equilibrium.

3.1 Problem Formulation and Solution

The game takes place on a graph with undirected edges. One node is designated as the origin, and another node as the destination. The k-ambush game is played as follows: Player 1 chooses a path from the origin to the destination. Player 2 simultaneously chooses k nodes at which to prepare ambushes. The outcome of an ambush at node i is equal to α_i if Player 1's path passes through node i, or 0 otherwise. (α_i models the fact that some nodes, by virtue of their geography or proximity to safe zones, are

more vulnerable to ambush than others. α_i will typically be zero at the origin and destination, since those two locations are likely to be heavily defended.)

Let $a_j, j = 1...k$ be the ambush nodes chosen by Player 2, and let x_i equal 1 if Player 1's path includes node i and 0 otherwise. Then the outcome of the game is given by:

$$V = \sum_{j=1}^{k} \alpha_{a_j} x_{a_j} \tag{3.1}$$

Given a directed graph $G = (\mathcal{N}, \mathcal{E})$, origin and destination nodes n_o and n_d, and ambush outcome values α_i for each node, the problem is to find Player 1's optimal mixed strategy for choosing among all paths between origin and destination, such that the expected outcome of the game is minimized.

Key assumptions in this game include:

- Both players have complete information about the environment, including the origin and destination nodes.
- Both Player 1 and Player 2 must choose their actions before the game begins. After the game has begun, there is no opportunity for Player 1 to alter his path or Player 2 to move his ambush sites.
- If Player 1 is ambushed more than once along his route, then the overall game outcome is equal to the sum of the outcomes of all ambushes.
- All ambushes take place at nodes (i.e. street intersections) rather than along edges.
- Player 2 is intelligent and will use the optimal strategy to place his ambushes. Therefore, Player 1's optimal strategy is the minimax strategy.

The assumption that Player 2 will place his ambushes at street intersections is not strictly necessary, but it does simplify some of the calculations. The approach described in this chapter minimizes the maximum weighted probability of Player

2 arriving at any k feasible ambush sites. Regardless of whether those sites are nodes or edges, the objective is still to minimize the maximum of some set of linear combinations of flow variables, and therefore this assumption could be dropped if required.

As mentioned earlier, this game can be solved by setting up a matrix with rows corresponding to all possible paths from origin to destination, and columns corresponding to all possible ambush strategies. However, a significantly more efficient formulation is to represent the game as a network flow optimization problem, with the flow along an edge equal to the probability of Player 1 traversing that edge.

3.1.1 1-Ambush Case

Let p_{ij} be the probability of Player 1 traversing the edge from node i to node j during his journey from origin to destination. Let q_i be the probability of Player 2 choosing node i at which to prepare his ambush. Then the expected outcome of the game is given by:

$$V = \sum_{j=1}^{|\mathcal{N}|} \left(\sum_{i|(i,j)\in\mathcal{E}} p_{ij} \right) \alpha_j q_j \tag{3.2}$$

i.e. take the probability of Player 1 arriving at node j, multiply by the probability of Player 2 preparing an ambush at that node, multiply again by the outcome of a successful ambush at that node, then sum across all nodes in the graph.

If both players pursue their optimal strategies, then the expected outcome is:

$$V^* = \min_P \max_Q V = \max_Q \min_P V \tag{3.3}$$

where P and Q are the sets of valid strategies for Players 1 and 2 respectively. The above equation is true by the minimax theorem, which holds for a finite, zero-sum, two-person game.

Let $\mathbf{p}$ denote the vector of probabilities p_{ij} for every edge $(i,j) \in \mathcal{E}$. To determine the optimal strategy for Player 1 (i.e. $\mathbf{p}^*$), we can use the fact that V^* is the minimax

of V, and that $q_i \leq 1 \; \forall \; i$. Player 2 can always maximize V by choosing the node for which the probability of Player 1 passing through that node weighted (i.e. multiplied) by the α value of that node is maximum. Therefore, Player 1's optimal solution is to minimize that product across all nodes:

$$\mathbf{p}^* = \arg\min_P \left\{ \max_j \left(\sum_{i|(i,j)\in\mathcal{E}} p_{ij} \right) \alpha_j \right\} \tag{3.4}$$

The above equation simply means that Player 1's optimal strategy is to minimize the maximum probability (weighted by α_j) of passing through any node j.

Because $\mathbf{p}$ is a flow vector, it must also satisfy network flow constraints (i.e. flow into a node equals flow out of the node):

$$\sum_{i|(i,j)\in\mathcal{E}} p_{ij} = \sum_{k|(j,k)\in E} p_{jk} \quad \forall \; j \; \setminus \{n_o, n_d\} \tag{3.5}$$

$$\sum_{k|(n_o,k)\in\mathcal{E}} p_{n_o k} = 1 \tag{3.6}$$

$$\sum_{i|(i,n_d)\in E} p_{i n_d} = 1 \tag{3.7}$$

And since each p_{ij} represents a probability:

$$p_{ij} \geq 0 \quad \forall \; (i,j) \in \mathcal{E} \tag{3.8}$$

Note that the upper bound of 1 for each p_{ij} is automatically enforced by this constraint and the fact that the net flow in to the graph is equal to 1.

This minimax problem can be formulated as a linear program by introducing a new variable z, which is a function of $\mathbf{p}$. The LP should force z to equal the maximum flow (weighted by α_i) into any node:

$$z(\mathbf{p}) \leftarrow \max_j \left(\sum_{i|(i,j)\in\mathcal{E}} p_{ij} \right) \alpha_j \tag{3.9}$$

We wish to find the $\mathbf{p}$ that will minimize z. The linear program is then:

$$
\begin{aligned}
\text{minimize} \quad & z \\
\text{subject to} \quad z &\geq \sum_{i|(i,j)\in\mathcal{E}} p_{ij}\alpha_j \quad \forall\, j \in \mathcal{N} \\
\sum_{i|(i,j)\in\mathcal{E}} p_{ij} &= \sum_{k|(j,k)\in\mathcal{E}} p_{jk} \quad \forall\, j \;\setminus \{n_o, n_d\} \\
\sum_{k|(n_o,k)\in\mathcal{E}} p_{n_o k} &= 1 \\
\sum_{i|(i,n_d)\in\mathcal{E}} p_{i n_d} &= 1 \\
p_{ij} &\geq 0 \quad \forall\, (i,j) \in \mathcal{E}
\end{aligned}
\tag{3.10}
$$

The LP can be written more concisely as:

$$
\begin{aligned}
\text{minimize} \quad & z \\
\text{subject to} \quad \mathbf{Dp} - \mathbf{1}z &\leq 0 \\
\mathbf{Ap} &= \mathbf{b} \\
\mathbf{p} &\geq 0
\end{aligned}
\tag{3.11}
$$

where the first set of constraints forces z to equal the maximum weighted flow into any node, the second set of constraints is the network flow conservation equations, and the third set of constraints prevents negative flows.

Solving this linear program will generate for each edge p^*_{ij}, which represents the probability of Player 1 traversing edge (i, j) on any given route if he follows his optimal strategy. p^*_{ij} is proportional but not equal to the transitional probability, which is the probability of moving from node i to node j given that he is already at node i. To generate a random path under strategy $\mathbf{p}^*$, at each node i Player 1 must normalize the outgoing probabilities to 1 before choosing the next node j based on these probabilities. (See Algorithm 3.3 for details.)

3.1.2 k-Ambush Case

In the case where Player 2 has the resources to prepare ambushes in k different locations, the problem is now to minimize the flow (weighted by α_i) into any *set* of k nodes, as opposed to minimizing the flow into any one node. Instead of having one inequality constraint for every node, there is now one inequality constraint for every set of k nodes, for a total of $\mathrm{C}^k_{|\mathcal{N}|}$ constraints. The flow constraints remain the same as in the 1-ambush case.

Let $N_k^1, N_k^2, \ldots$ be all subsets of k distinct nodes within $\mathcal{N}$:

$$N_k^l = \{(n_1^l, n_2^l, \ldots, n_k^l) \mid n_i^l \in \mathcal{N} \; \forall \; i \; , \; n_i^l \neq n_j^l \; \forall \; i, j\} \tag{3.12}$$

Using the same variable z as before, the constraints on z for the k-ambush case are then:

$$z \; \geq \; \sum_{j \in N_k^l} \left(\alpha_j \sum_{i|(i,j)\in\mathcal{E}} p_{ij} \right) \quad \forall \; N_k^l \subset \mathcal{N} \tag{3.13}$$

The constraints in (3.13) are still of the same form as those in (3.9) (i.e. $\mathbf{Dp} - \mathbf{1}z \leq 0$). The flow constraints are unchanged. Therefore, the LP for the k-ambush case is equivalent to the LP for the 1-ambush case, except the matrix $\mathbf{D}$ is replaced with a larger matrix $\mathbf{D}_k$:

$$\begin{aligned} \text{minimize} \quad & z & & \\ \text{subject to} \quad & \mathbf{D}_k\mathbf{p} - \mathbf{1}z & \leq & \; 0 \\ & \mathbf{Ap} & = & \; \mathbf{b} \\ & \mathbf{p} & \geq & \; 0 \end{aligned} \tag{3.14}$$

In general, for the k-ambush case, the number of decision variables is equal to the number of directed edges plus one, and the number of constraints is equal to $\mathrm{C}^k_{|\mathcal{N}|} + |\mathcal{N}|$.

3.1.3 Example

A simple example graph is shown in Figure 3-1. Node 1 is the origin and node 8 is the destination. In the equations below, these nodes are not included as possible ambush sites, because in a real-life situation they are likely to be heavily defended "safe zones".

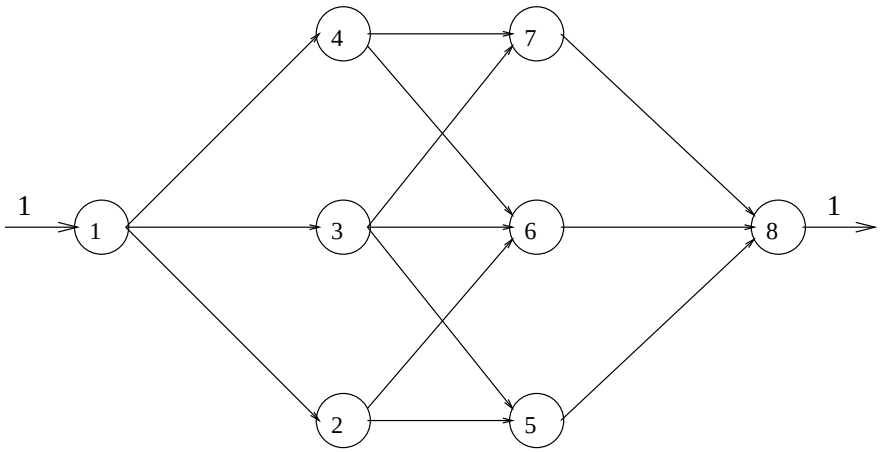

Figure 3-1: Example graph

For the 1-ambush case, the corresponding LP is:

$$
\begin{aligned}
\min \quad & z & & \\
\text{subject to} \quad & z & \geq & \alpha_2 p_{12} \\
& z & \geq & \alpha_3 p_{13} \\
& z & \geq & \alpha_4 p_{14} \\
& z & \geq & \alpha_5 (p_{25} + p_{35}) \\
& z & \geq & \alpha_6 (p_{26} + p_{36} + p_{46}) \\
& z & \geq & \alpha_7 (p_{37} + p_{47}) \\
1 - p_{12} - p_{13} - p_{14} & & = & 0 \\
p_{12} - p_{25} - p_{26} & & = & 0 \\
p_{13} - p_{35} - p_{36} - p_{37} & & = & 0 \\
p_{14} - p_{46} - p_{47} & & = & 0 \\
p_{25} + p_{35} - p_{58} & & = & 0 \\
p_{26} + p_{36} + p_{46} - p_{68} & & = & 0 \\
p_{37} + p_{47} - p_{78} & & = & 0 \\
p_{58} + p_{68} + p_{78} - 1 & & = & 0 \\
p_{ij} & & \geq & 0
\end{aligned}
$$

In the form of equation (3.14), the corresponding matrices and vectors are:

$$\mathbf{p} = [\ p_{12} \quad p_{13} \quad p_{14} \quad p_{25} \quad p_{26} \quad p_{35} \quad p_{36} \quad p_{37} \quad p_{46} \quad p_{47} \quad p_{58} \quad p_{68} \quad p_{78}\ ]^T$$

$$\mathbf{D}_1 = \begin{bmatrix} \alpha_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_5 & 0 & \alpha_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha_6 & 0 & \alpha_6 & 0 & \alpha_6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha_7 & 0 & \alpha_7 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{b} = [\ -1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1\ ]^T$$

The optimal solution to this LP gives Player 1's optimal strategy $\mathbf{p}^*$ for navigating from node 1 to node 8, and the expected game value z^* if Player 2 pursues his optimal strategy. Figure 3-2 shows an optimal solution in the case where $\alpha_i = 1$ for all nodes. All edges in the figure have probability 1/3, and the remaining edges, such as edge (2,6), have been assigned zero probability. The expected game value is 1/3 because under this strategy Player 1 has exactly a 1/3 probability of arriving at any of nodes 2 through 8.

Note that the solution in Figure 3-2 is not unique. For example, if p_{37} and p_{46} are set to zero, and p_{36} and p_{47} to 1/3, the expected game value will still equal 1/3.

In the 2-ambush case, the $\mathbf{A}$ matrix and $\mathbf{b}$ vector remain the same, since the network flow constraints do not change, but the objective is to minimize the maximum

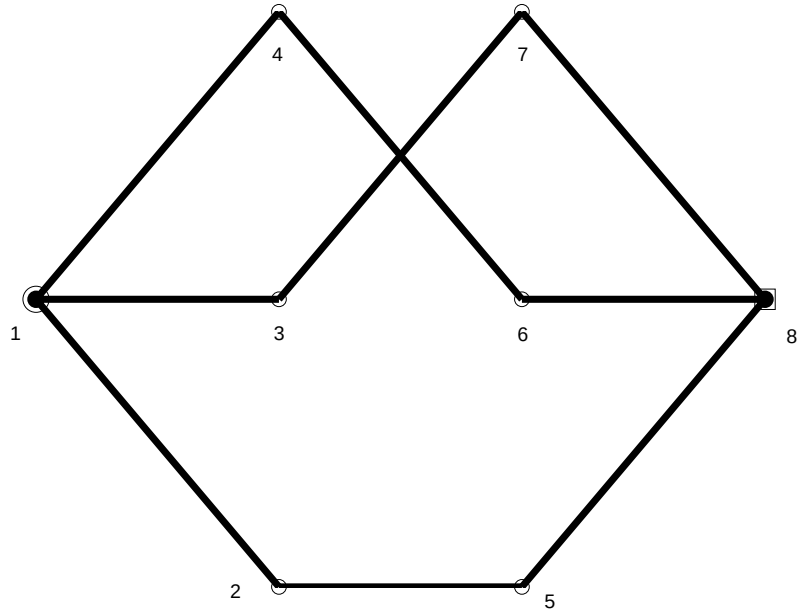

Figure 3-2: Example graph solution for 1-ambush case

α-weighted flow into any set of 2 nodes. The set of constraints involving z is now:

$$\begin{aligned} z &\geq \alpha_2 p_{12} + \alpha_3 p_{13} \\ z &\geq \alpha_2 p_{12} + \alpha_4 p_{14} \\ z &\geq \alpha_2 p_{12} + \alpha_5 (p_{25} + p_{35}) \\ z &\geq \alpha_2 p_{12} + \alpha_6 (p_{26} + p_{36} + p_{46}) \\ &\text{etc.} \end{aligned}$$

There are 15 possible ways for Player 2 to choose 2 ambush nodes out of the six possible ambush nodes (omitting the origin and destination), so there are a total of 15 such constraints in the 2-ambush problem. $\mathbf{D}_2$ is then a 15x13 matrix.

3.1.4 Presence of Circulations in Player 1's Optimal Strategy

A *cycle* in a graph is a path containing nodes $i_1, \ldots, i_t$ for which any 2 consecutive nodes is connected by an edge, and $i_t = i_1$. A *circulation* is a flow that is nonzero for all edges of a cycle and zero for other edges (i.e. a flow that "circulates" within the graph). The cycle and its circulation are *directed* if all the edges in the cycle are forward edges.

In a regular network flow linear optimization problem, a basic solution can not contain any circulations (e.g. see Chapter 7 in [3]). However, that is not the case for the LP in (3.14), which contains an extra non-flow variable z. The optimal strategy for Player 1 will almost always contain at least one undirected circulation (otherwise the strategy would be a pure strategy with only one path from origin to destination), but it is also possible for directed circulations to appear in the solution.

Directed circulations can be consistent with an optimal strategy because the objective is to minimize the maximum weighted flow into any set of k nodes. A directed circulation will add flow to the set of nodes that compose the cycle, but as long as the total weighted flow into any one of these nodes does not exceed z^*, then the circulation does not affect the optimal value.

Although the expected outcome of the game (z^*) is not affected by the presence of a directed circulation, it is still undesirable from Player 1's point of view for two reasons. The first is a practical reason: when navigating from origin to destination by adhering to the edge probabilities, the agent has the potential to loop around a cycle one or more times, which is a complete waste of time. Secondly, the LP in (3.14) assumes the worst case for Player 1: that Player 2 will follow his optimal strategy and place ambushes at nodes where the incoming weighted flow equals z^*. However, if Player 2 follows a suboptimal strategy and places ambushes at nodes along the cycle, then the outcome will be unnecessarily bad for Player 1. Effectively, Player 1 gains nothing but has the potential to lower his payoff if his strategy contains directed circulations.

Fortunately, the offending circulation can simply be excised from the optimal

solution, as shown in the following theorem:

Theorem 3.1. *Suppose* $\mathbf{p}^*$ *is an optimal strategy for Player 1 in the VIP transport game, as defined in (3.14), and* $\mathbf{p}^*$ *contains a directed circulation* $\mathbf{p}^c$*. Then* $(\mathbf{p}^* - \mathbf{p}^c)$ *is also an optimal strategy for Player 1.*

Proof. Assume that the expected game outcome associated with strategy $\mathbf{p}^*$ is z^*. It is sufficient to show that $(\mathbf{p}^* - \mathbf{p}^c, z^*)$ is a feasible solution to (3.14):

1. *Positive flow*: $\mathbf{p}^*$ is a positive flow, and $\mathbf{p}^c$ is a positive flow contained within $\mathbf{p}^*$, therefore $\mathbf{p}^* > \mathbf{p}^c > 0$ and $(\mathbf{p}^* - \mathbf{p}^c) > 0$.

2. *Flow conservation*: Since $\mathbf{p}^c$ is a circulation, the flow into any node equals the flow out of that node, and so it satisfies $\mathbf{A}\mathbf{p}^c = 0$. We also have $\mathbf{A}\mathbf{p}^* = 0$ since $\mathbf{p}^*$ is a feasible flow. Therefore $\mathbf{A}(\mathbf{p}^* - \mathbf{p}^c) = 0$.

3. *Optimality*: The coefficients of $\mathbf{D}_k$ are either zero or equal to the α_i parameters, which are positive. Since $\mathbf{p}^c$ is a directed circulation, $\mathbf{p}^c > 0$ and therefore $\mathbf{D}_k\mathbf{p}^c > 0$. Finally, since $(\mathbf{p}^*, z^*)$ is feasible, we have $\mathbf{D}_k\mathbf{p}^* - \mathbf{1}z^* \leq 0$ and therefore $\mathbf{D}_k(\mathbf{p}^* - \mathbf{p}^c) - \mathbf{1}z^* \leq 0$.

Since $(\mathbf{p}^* - \mathbf{p}^c, z^*)$ is a feasible solution to (3.14), and z^* is known to be the optimal value of the objective function in (3.14), therefore $(\mathbf{p}^* - \mathbf{p}^c)$ is also an optimal strategy. □

Although it is trivial to remove the circulation, it may not be as simple to find the circulation in the first place. Instead of developing an algorithm to locate and remove circulations from a flow vector, it is more advisable to slightly modify the LP so that directed circulations will not appear in the solution. The objective function can be modified by adding a small cost coefficient δ for every flow variable p_{ij}, as shown in (3.15).

$$\begin{aligned} \min \quad & \delta \sum_{(i,j)\in\mathcal{E}} p_{ij} + z \\ \text{subject to} \quad & \mathbf{D}_k\mathbf{p} - \mathbf{1}z \leq 0 \\ & \mathbf{A}\mathbf{p} = \mathbf{b} \\ & \mathbf{p} \geq 0 \end{aligned} \tag{3.15}$$

For sufficiently small $\delta > 0$, the modified LP will yield the same solution but without directed circulations. If δ is chosen too large such that the LP yields a suboptimal solution $(\mathbf{p}, z)$, then we can be sure that $z - z^* < \delta \cdot |\mathcal{E}|$.

3.1.5 Dependency of the Expected Game Outcome on k

Given $(\mathbf{p}^*_{k_1}, z^*_{k_1})$ as an optimal solution to the k_1-ambush game, is there anything that can be said about the optimal solution to the k_2-ambush game, for $k_2 > k_1$? This question may be relevant because the computation time required to determine the solution increases exponentially with k.

Player 2's best strategy for the k_2-ambush game is obviously to place ambushes at the k_2 nodes with the greatest α-weighted inflow. Suppose Player 1 pursues strategy $\mathbf{p}^*_{k_1}$ for the k_2-ambush game. Then the worst case for Player 1 will occur if, under this strategy, there are at least k_2 nodes which each has an inflow of $z^*_{k_1}/k_1$. An upper bound on $z^*_{k_2}$ is therefore:

$$z^*_{k_2} \leq \frac{k_2}{k_1} z^*_{k_1} \tag{3.16}$$

3.1.6 Variation for Multiple Agents

This network flow approach can be extended to compute the optimal strategies for several VIPs who must simultaneously journey in separate vehicles through the same city. These VIPs may or may not have different origins and destinations. The objective is to minimize the max α-weighted flow (summed across all agents) into any

set of k nodes. We still assume that the outcome of the game is equal to the sum of outcomes of each ambush.

With each VIP in a separate vehicle, the network flow problem becomes a *multi-commodity flow* problem. Each of the n VIP agents has its own strategy: $\mathbf{p}^1, \ldots, \mathbf{p}^n$. The new LP is:

$$\begin{aligned} & \text{minimize} \quad z & & \\ \text{subject to} \quad & \mathbf{D}_k\mathbf{p}^1 + \mathbf{D}_k\mathbf{p}^2 + \cdots + \mathbf{D}_k\mathbf{p}^n - \mathbf{1}z \;\leq\; 0 & & \\ & \mathbf{A}\mathbf{p}^1 \;=\; \mathbf{b}^1 & & \\ & \mathbf{A}\mathbf{p}^2 \;=\; \mathbf{b}^2 & & \\ & \vdots & & \\ & \mathbf{A}\mathbf{p}^n \;=\; \mathbf{b}^n & & \\ & \mathbf{p} \;\geq\; 0 & & \end{aligned} \tag{3.17}$$

This problem has a structure that is particularly suited to Dantzig-Wolfe decomposition methods. The problem can be reformulated as a master problem and n subproblems. The primary advantage of this method is that it significantly reduces the storage requirements compared to applying the regular simplex method to the original problem. For a more detailed discussion of Dantzig-Wolfe decomposition, see Chapter 6 in [3].

3.1.7 Complexity

For a linear program in standard form ($\min \mathbf{x}$ s.t. $\mathbf{A}\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq 0$), the worst-case number of computations required to perform a single iteration of the simplex algorithm is $\mathcal{O}(mn)$, where m is the number of equality constraints and n is the number of decision variables [3].

The linear program in (3.11) that solves the 1-ambush problem has decision variables $\mathbf{p}$ and z for a total of $|\mathcal{E}| + 1$ decision variables. There are $|\mathcal{N}|$ inequality

constraints (from $\mathbf{Dp} - \mathbf{1}z \leq 0$) and $|\mathcal{N}|$ equality constraints (from $\mathbf{Ap} = \mathbf{b}$). To convert the LP to standard form, a slack variable must be added for each of the inequality constraints, and therefore the standard form LP will have $|\mathcal{E}| + |\mathcal{N}| + 1$ decision variables and $2|\mathcal{N}|$ equality constraints. The worst-case computational requirement for a single iteration of the simplex method is then $\mathcal{O}(|\mathcal{E}||\mathcal{N}| + |\mathcal{N}|^2)$, or simply $\mathcal{O}(|\mathcal{E}||\mathcal{N}|)$ since $|\mathcal{E}| > |\mathcal{N}|$ for a typical graph.

In comparison, if the naive implementation in (2.1) is used, the number of decision variables is equal to $|\mathcal{R}| + |\mathcal{N}|$, where $\mathcal{R}$ is the set of all possible paths from the origin to the destination. With $\mathcal{N}$ equality constraints in standard form, the worst-case computational requirement is $\mathcal{O}(|\mathcal{R}||\mathcal{N}|)$. Since $|\mathcal{R}| >> |\mathcal{E}|$ for a typical graph, the network flow approach will in most cases yield the optimal solution in significantly less time.

For the k-ambush case in (3.14), the results are similar except the number of inequality constraints in the original LP increases from $|\mathcal{N}|$ to $C_k^{|\mathcal{N}|}$. After adding slack variables, the number of decision variables is $C_k^{|\mathcal{N}|} + |\mathcal{E}| + 1$, where the first term generally dominates. For $k << |\mathcal{N}|$ (again, a typical scenario), $C_k^{|\mathcal{N}|} \sim |\mathcal{N}|^k$ and therefore the worst-case computational requirement for a single iteration of simplex is $\mathcal{O}(|\mathcal{N}|^{2k})$.

3.2 Implementation

In this section, we discuss how to implement the methods described in the previous section to obtain optimal routing solutions for real-life ambush scenarios.

3.2.1 Modeling

The following process is used to model a VIP transport game scenario:

1. Acquire a road map of the area of interest.

2. Identify roads that are traversable by the agent (e.g. sealed roads only, roads greater than a certain width, etc.).

3. Identify the intersections of these roads. Associate each intersection with a node and record the coordinates of each node. Denote the set of nodes as $\mathcal{N}$.

4. Identify road segments that connect nodes. Associate each segment with an edge. Denote the set of edges as $\mathcal{E}$.

5. Identify the origin and destination nodes. Denote these nodes as n_o and n_d respectively.

6. Estimate the expected ambush outcome at each node. Denote the value associated with node i as α_i. (In the absence of useful information to make this estimate, α_i can be set by default to a value of 0 for the origin and destination nodes, and 1 for all other nodes.)

7. Estimate the number of ambushes that the enemy is capable of preparing. Denote this value as k.

Having established the parameters of the problem, the decision variables need to be specified. Using the network flow formulation, the decision variables are the probabilities p_{ij} associated with all edges $(i, j) \in \mathcal{E}$. p_{ij} denotes the probability of the VIP traveling from node i to node j in any journey from the origin to the destination. These edge probabilities are the elements of the vector $\mathbf{p}$.

3.2.2 Formulating the Constraints

The first step in formulating the constraints is to generate a list of all possible ambush strategies for Player 2 (i.e. the sets $\mathcal{N}_k^l$ in (3.12)). $\mathcal{N}_k^l \subset \mathcal{N}$ are simply the combinations of size k in the set of nodes $\mathcal{N}$, and there exist many well-known algorithms for generating combinations from an arbitrary set.

The second step is to populate the $\mathbf{D}_k$ matrix using Algorithm 3.1.

Algorithm 3.1 Populating the $\mathbf{D}_k$ matrix

$\mathbf{D}_k \leftarrow \mathbf{0}$
for all $\mathcal{N}_k^l \subset \mathcal{N}$ **do**
  for $n \in \mathcal{N}_k^l$ **do**
    for all $j \mid (j, n) \in \mathcal{E}$ **do**
      $D_k(i, j) \leftarrow D_k(i, j) + \alpha_n$
    end for
  end for
end for

The third step is to populate the $\mathbf{A}$ matrix using Algorithm 3.2.

Algorithm 3.2 Populating the $\mathbf{A}$ matrix

$\mathbf{A} \leftarrow \mathbf{0}$
for all $(n_1, n_2) \in \mathcal{E}$ **do**
  $A(n_1, j) \leftarrow -1$
  $A(n_2, j) \leftarrow 1$
end for

Finally, the $\mathbf{b}$ vector is simply given by:

$$b_i = \begin{cases} -1 & \text{if } i = n_o \\ 1 & \text{if } i = n_d \\ 0 & \text{otherwise} \end{cases}$$

3.2.3 Solving the LP

Equation (3.14) can be solved using any implementation of the simplex method (e.g. revised simplex, full tableau). Although the problem is based on a network flow approach, the extra non-flow decision variable z ensures that it is not a typical network flow LP, and so algorithms developed specifically for network flow problems (e.g. the network simplex algorithm) can not necessarily be applied here.

Empirically, the more general simplex method seems to solve the problem quite efficiently. For details, see Section 3.3.

3.2.4 Generating a Random Path

The solution to the LP includes the optimal mixed strategy $\mathbf{p}^*$ which contains the optimal probability p^*_{ij} for every edge $(i,j) \in \mathcal{E}$. These probabilities are the overall probabilities across all journeys for traversing a particular edge, and so the sum of probabilities of edges out of a particular node will not necessarily sum to 1. Therefore, when generating a random path, at each node the probabilities of outgoing edges must be normalized to one before an edge is chosen.

Algorithm 3.3 illustrates this procedure.

Algorithm 3.3 Generating a Random Path

```
/* Let r denote the ordered list of nodes composing the path */
n ← 1
r_1 ← n_o

while r_n ≠ n_d do

  /* Normalize the probabilities for outgoing edges */
  N ← j | (r_n, j) ∈ E
  P_sum ← 0
  for all j ∈ N do
    P_sum ← P_sum + p*_{r_n j}
  end for
  for all j ∈ N do
    p̃*_{r_n j} ← p*_{r_n j} / P_sum
  end for

  /* Randomly choose r_{n+1} based on normalized probabilities */
  x ← rand[0, 1]
  s ← 0
  i ← 1
  while s < x do
    s ← s + p̃*_{r_n N_i}
    i ← i + 1
  end while
  r_{n+1} ← N_i
  n ← n + 1

end while
```

3.3 Cambridge Scenario Application

The methods described in this chapter were applied to a number of different scenarios set in the city of Cambridge, MA. Cambridge is a good test case because its highly irregular road network makes the task of determining the optimal routing strategy quite difficult. This example also serves to demonstrate that these solution methods are practical and efficient enough to run on a regular desktop or laptop computer.

A program was written to input the parameters of the problem (the graph, origin and destination nodes, and ambush outcome values for each node) and output the optimal routing strategy. The two computationally intensive portions of the program are formulating the constraints and solving the LP. The first is implemented in Matlab, and the second is implemented in C using GLPK[1] simplex routines. The program was run under Windows XP on a Pentium 4 2.20 GHz desktop PC with 512 MB of RAM.

3.3.1 Environment Model

A map of Cambridge was converted to a a graph containing 50 nodes and 91 edges, as shown in Figure 3-3. The choice of roads to include was somewhat arbitrary in this case, but mainly limited to major roads. In practice, operational reasons would rule out certain roads due to width, load-carrying capacity, obstructions, roadwork, etc.

Most of these roads are two-way roads, and therefore the corresponding edges are valid in both directions (e.g. (36,37) and (37,36) are both included in $\mathcal{E}$). A few roads are one-way—for example, $(16, 24) \in \mathcal{E}$ but $(24, 16) \notin \mathcal{E}$.

The ambush outcome at a node was taken to be the smallest of the distances from that node to any origin or destination nodes. In that way, the cost of an ambush at an origin or destination was considered to be zero, which accurately reflects the fact that those locations are likely to be heavily defended and safe against ambush. Also,

[1] See http://www.gnu.org/software/glpk/glpk.html for further information on this excellent open-source software package.

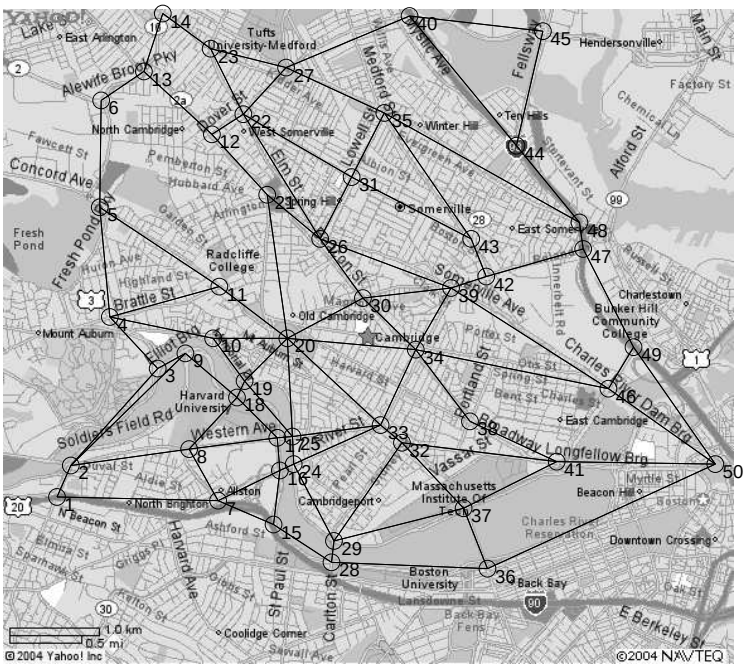

Figure 3-3: Map of Cambridge with overlaid graph

as the agent moves further away from these locations, he is more vulnerable if he is ambushed.

3.3.2 Single-Agent Scenarios

In the single-agent scenarios, Player 1 must navigate through the city of Cambridge from Fresh Pond to downtown Boston. The origin node is number 5, and the destination node is number 50. Optimal routing strategies were found for the 1-, 2-, and 3-ambush cases.

The times taken to formulate and solve the 1-, 2-, and 3-ambush cases, together with the expected game outcomes, are shown in Table 3.1. Note that the upper bound in (3.16) is satisfied: $z_3^* < \frac{3}{2} z_2^*$, $z_3^* < 3z_1^*$, and $z_2^* = 2z_1^*$.

The computation time increases exponentially with the number of ambushes. Fortunately, in a typical scenario it is unlikely that the enemy will have the resources to prepare a large number of different ambushes simultaneously.

Scenario	Constraint Formulation	LP Solution	Expected Outcome
1-ambush	0.059 s	0.015 s	46.7
2-ambush	1.12 s	0.11 s	93.5
3-ambush	25.41 s	7.81 s	138.3

Table 3.1: Results for 1-, 2-, and 3-ambush scenarios

Figure 3-4 shows an optimal strategy for Player 1 for each of the 3 cases. In general, the optimal strategy is a result of a tradeoff between (1) avoiding dangerous, high-α nodes (i.e. those in the NW and SE corners of the map), and (2) incorporating as many distinct routes as possible, to reduce the probability of arrival at any individual node.

In this scenario, the optimal strategy for the 1-ambush game turns out to be optimal for the 2-ambush game as well, and the expected game outcome for the 2-ambush game is exactly double that for the 1-ambush game. However, for the 3-ambush game, a slightly different strategy is optimal and the expected game outcome is slightly lower than three times the expected outcome for the 1-ambush game.

The structure of the LP in (3.14) can explain why $\mathbf{p}_2^* = \mathbf{p}_1^*$ and $z_2^* = 2z_1^*$. Note that this LP minimizes the maximum α-weighted flow into any set of k nodes, and let w_i equal the α-weighted flow into node i. In the 1-ambush case with strategy $\mathbf{p}_1^*$, there is at least one node i for which $w_i = z_1^*$, but there can easily exist another node j for which $w_j = z_1^*$. In the 2-ambush case, unless there exists a strategy for which the two greatest α-weighted inflows sum to less than $2z_1^*$, then the optimal strategy will simply be $\mathbf{p}_1^*$, which is indeed the case for this example. For the 3-ambush case, however, there clearly exists a strategy for which the three greatest α-weighted inflows sum to less than $3z_1^*$, and therefore $\mathbf{p}_3^* \neq \mathbf{p}_1^*$.

It is important to note that the optimal strategies are not necessarily unique. The objective function is determined only by those k nodes with the greatest α-weighted inflow. The flow into the remaining $|\mathcal{N}| - k$ nodes can often be rearranged without the inflow into any of them exceeding w_k, and also without affecting the inflow into any of the critical k nodes. In that case, all such rearrangements would represent optimal solutions.

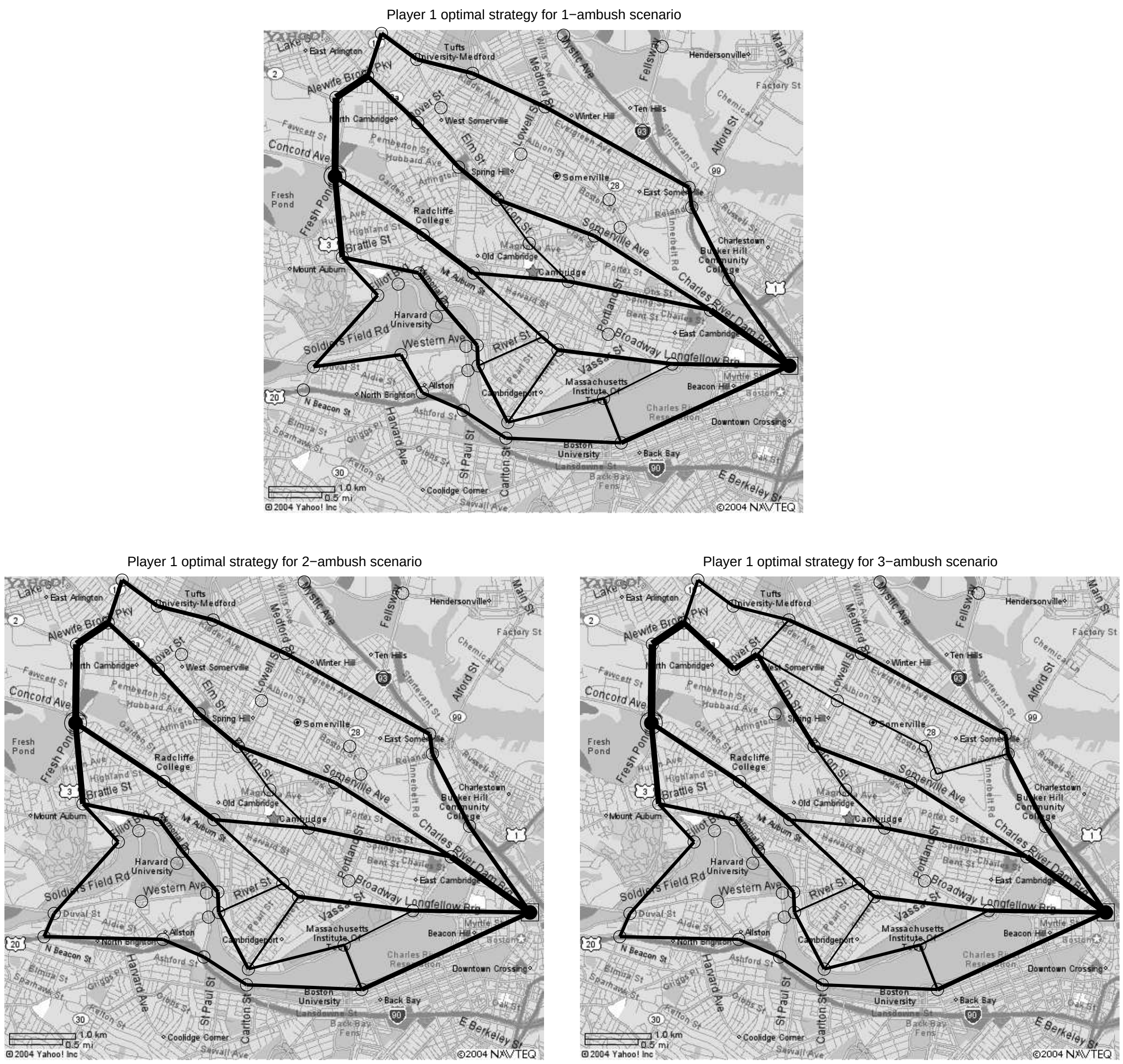

Figure 3-4: Player 1 optimal strategy for 1-, 2-, and 3-ambush scenarios. Darker lines correspond to higher probability edges.

3.3.3 Multiple-Agent Scenario

This scenario includes three VIPs with separate vehicles, origins, and destinations. The enemy has the capability to prepare one ambush. The goal is to find the optimal routing strategy for each VIP such that the α-weighted probability of a successful ambush on any VIP is minimized. The origin nodes for the three VIPs were 5, 1, and 28 respectively, and the destination nodes were 50, 45, and 27 respectively (see Figure 3-3).

Figure 3-5 shows the optimal routing strategy for each VIP. Computation time was 0.05 seconds for formulating the constraints, and 0.06 seconds for solving the LP. The expected outcome of the game is 92.5.

Figure 3-6 is a combination of the three plots in Figure 3-5. The thickness of each line is proportional to the sum of the probabilities of any of the three agents traversing that edge. The optimal strategies for each of the agents combine effectively to distribute their routing over most of the graph, and prevent the agents from arriving together at any one node with too high a probability.

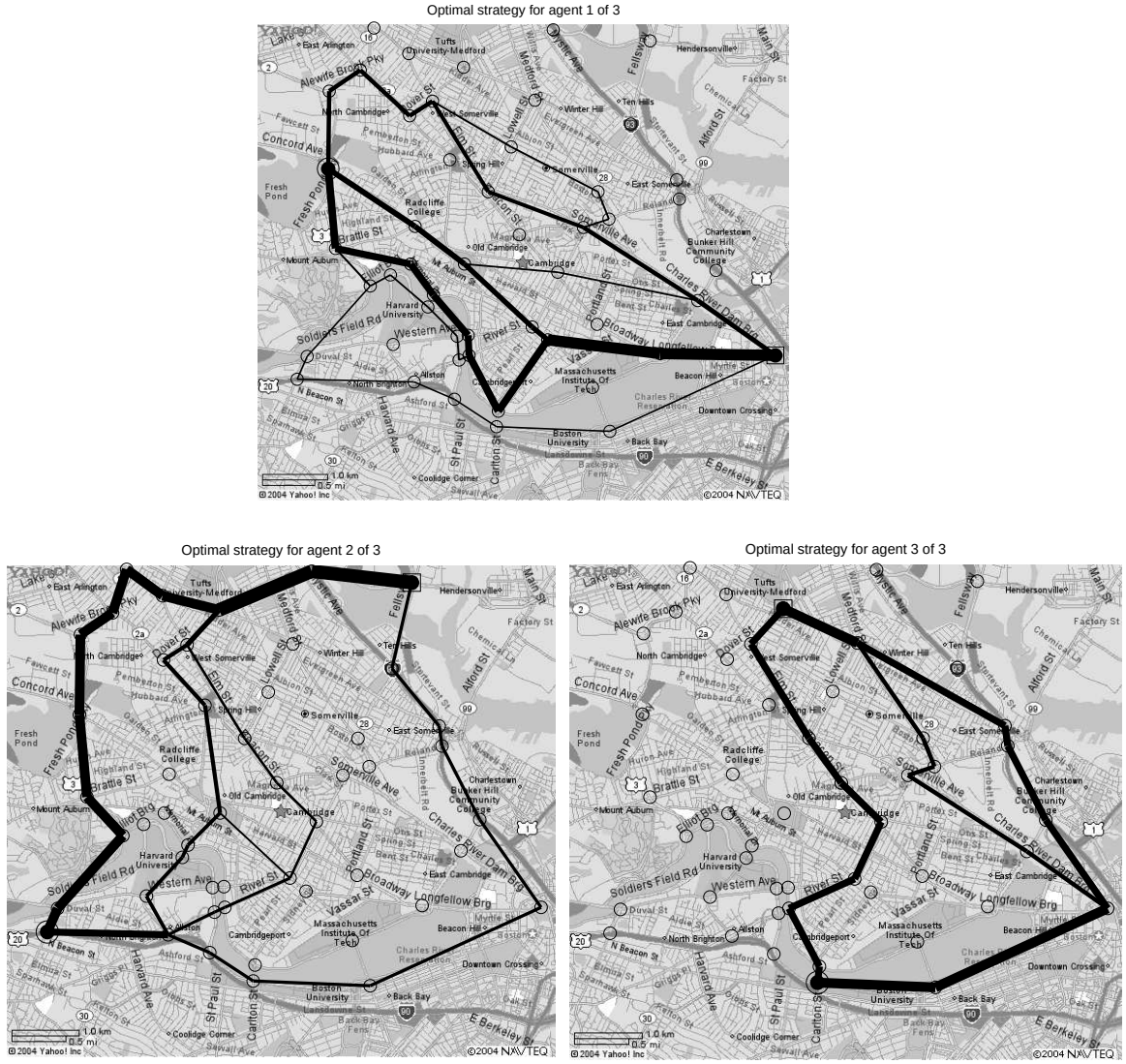

Figure 3-5: Optimal strategies for 3 agents.
Darker lines correspond to higher probability edges.

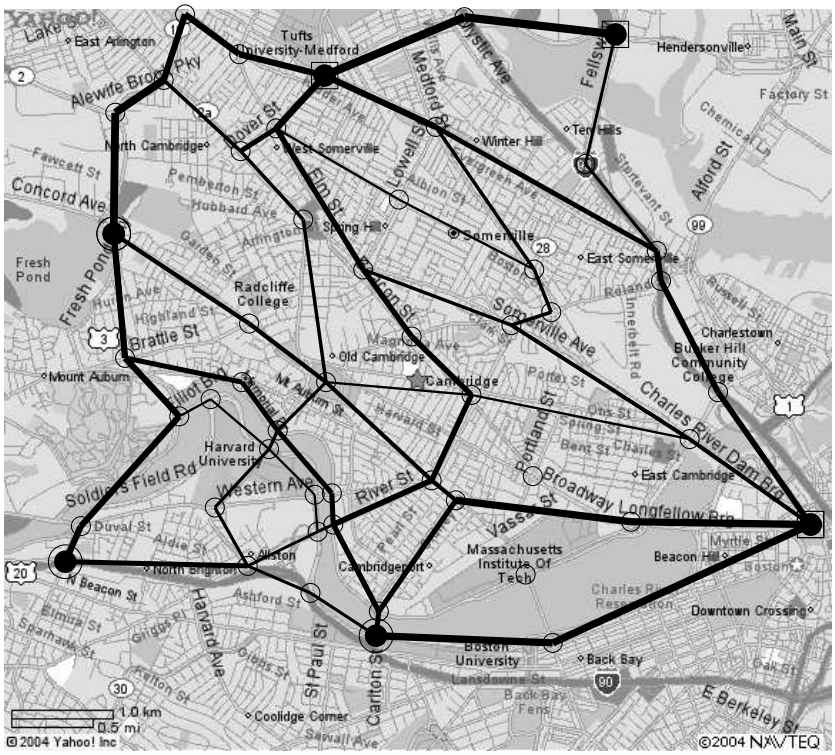

Figure 3-6: Combined optimal strategies for 3 agents, separate vehicles

Chapter 4

Multi-Stage Ambush Games

The VIP transport problem discussed in Chapter 3 is an example of a single-stage game, in which each player makes only one decision in the course of the game, and the decisions are made simultaneously at the outset of the game. Such a game would be an accurate representation of a scenario in which the hostile forces either cannot observe the VIP's location unless he reaches an ambush site, or they can observe the VIP's location but are unable to take advantage of the information (e.g. because the ambush site takes too much time to prepare, and must be done well ahead of time).

In this chapter, we discuss a variety of ambush games in which Player 2 can observe Player 1's position as Player 1 travels from origin to destination, and Player 2 has the ability to act on the information. We determine Player 1's optimal strategy under those circumstances.

We refer to these games as "multi-stage games". In this chapter, a "stage" refers to the set of decisions and actions taken every time Player 1 arrives at a new node. A new stage begins after Player 1 arrives at a new node, and ends with either an ambush or Player 1's safe arrival at the next node.

All games considered here take place on a similar graph as the game considered in Chapter 3. Given a directed graph $G = (\mathcal{N}, \mathcal{E})$, origin and destination nodes n_o and n_d, and ambush cost values α_i for each node, the problem is to find Player 1's optimal mixed strategy for choosing among all paths between origin and destination, such that the expected outcome of the game is minimized. The difference now is that

Player 2 does not have to choose his ambush sites at the outset of the game, but may incorporate information about Player 1's position en route to the destination before making his decision.

Because of this dynamic element, these multi-stage ambush games can not necessarily be easily solved using the network flow approach of Chapter 3. The limitation of the network flow approach is that it only concerns the absolute flow through an edge, and not the time at which the flow through the edge occurs, which is essential information in the dynamic game.

On the other hand, the multi-stage games can usually be solved one node at a time using a dynamic programming approach. Each stage of the game is in fact a game in itself, in which Player 1 chooses the next node to move towards and Player 2 chooses to place an ambush (in Games 1, 2, and 3) or bide his time (in Games 2 and 3). The sub-game at each node is relatively small and quick to solve.

To apply the dynamic programming approach, we assume the *principle of optimality* as defined in Chapter 7 in [3]. Suppose $\mathbf{P}^*_{ik}$ is the optimal routing strategy from node i to node k, and j is an intermediate node between i and k. Then the strategy dictated by $\mathbf{P}^*_{ik}$ between j and k is equivalent to the optimal routing strategy $\mathbf{P}^*_{jk}$ from j to k. In other words, if the optimal routing strategy from i to k includes an intermediate node j, then the part of the strategy from j to k is optimal in itself.

It should be noted that these multi-stage ambush games are very similar to an entire class of games that show up in a variety of different fields. One interesting example can be found in [6], which refers to the use of game theory to model competitive dancing. At each stage of a dance routine, the dancers select a dance move and their payoff increases if their chosen move is not guessed by their "opponent", the judges. In [5], a dynamic programming approach is actually used to determine the optimal mixed strategy for a choreographed dance routine.

4.1 Game 1

In this game, Player 2 can prepare one ambush at every stage. This game may be an appropriate model for a scenario in which the hostile forces (Player 2) are divided into units that are dispersed in a number of stages between the origin and destination, and each unit has one opportunity to intercept a convoy (Player 1) as it passes through the unit's stage. Whenever the convoy passes through a stage, its position is observed and transmitted to the unit in the next stage, and that unit then must decide the best location within the next stage to prepare an ambush.

4.1.1 Problem Formulation and Solution

Each stage of this game proceeds as follows:

1. Player 2 observes Player 1's position.

2. Player 1 chooses an adjacent node to move to.

3. Player 2 chooses an adjacent node at which to place an ambush.

If Player 1 and Player 2 choose different nodes, the outcome of that stage of the game is zero and the above process repeats at this new node. If Player 1 and Player 2 choose the same node j, there are two possibilities depending on the game rules. One possibility is that the game ends at that point, in which case the final game outcome is equal to α_j. The other possibility is that the game continues, but the game outcome is incremented by α_j.

Assume Player 1 is currently located at node i, and has the option to move to any one of nodes $j_1 \dots j_n$. Assume that we have already found Player 1's optimal strategy if he starts at each of these n nodes, and the expected value of the game (assuming each player follows his optimal strategy) starting from each node is $V_{j_1} \dots V_{j_n}$. Then the game matrix at node i is:

$$\mathbf{A} = \begin{bmatrix} \alpha_{j_1} & V_{j_1} & V_{j_1} & \dots & V_{j_1} \\ V_{j_2} & \alpha_{j_2} & V_{j_2} & \dots & V_{j_2} \\ V_{j_3} & V_{j_3} & \alpha_{j_3} & \dots & V_{j_3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ V_{j_n} & V_{j_n} & V_{j_n} & \dots & \alpha_{j_n} \end{bmatrix} \tag{4.1a}$$

where row x corresponds to Player 1's choice of node j_x to move to, and row y corresponds to Player 2's choice of node j_y to prepare an ambush.

If the game continues after a successful ambush, then the game matrix is:

$$\mathbf{A} = \begin{bmatrix} V_{j_1} + \alpha_{j_1} & V_{j_1} & V_{j_1} & \dots & V_{j_1} \\ V_{j_2} & V_{j_2} + \alpha_{j_2} & V_{j_2} & \dots & V_{j_2} \\ V_{j_3} & V_{j_3} & V_{j_3} + \alpha_{j_3} & \dots & V_{j_3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ V_{j_n} & V_{j_n} & V_{j_n} & \dots & V_{j_n} + \alpha_{j_n} \end{bmatrix} \tag{4.1b}$$

Once the game matrix is established, the optimal strategy at that point can easily be found by solving the linear program in (2.1). Although this method is inefficient for the large game matrices associated with the single-stage ambush game, in this case the size of the game matrix is limited by the number of edges leaving the node, and therefore the LP can be expected to yield a fast solution.

The solution to this linear program at node i gives the optimal probabilities p_{ij} for moving from node i to any adjacent node j, together with the expected game outcome V_i if the game begins at that node.

To determine Player 1's optimal strategy from the origin, a dynamic programming approach is used. The destination node is assigned a value of $V = 0$ (intuitively, if Player 1 starts at the destination, then there is no chance of being ambushed on the way there). Nodes adjacent to the destination will also have a value of 0. Working backwards from the destination node, the expected game value can be found at each node, together with the optimal strategy for moving from that node to adjacent nodes. This method is detailed in Algorithm 4.1.

Algorithm 4.1 Computing the optimal strategy for Game 1

```
/* U = set of unlabeled nodes */
U ← N \ n_d

/* L = set of labeled nodes */
L ← {n_d}

/* O(i) = set of nodes linked from node i */
O(i) = {j ∈ N | (i, j) ∈ E}

V_{n_d} = 0

while U ≠ ∅ do
  for all i ∈ U | O(i) ∈ L do
    Construct A from (4.1a) or (4.1b)
    Solve for V_i and p_i^* using (2.1)
    U ← U \ i
    L ← L ∪ {i}
  end for
end while
```

4.1.2 Complexity

The Game 1 solution requires a matrix game to be solved at every node of the graph from the destination back to the origin. A total of $|\mathcal{N}|$ linear programs must therefore be solved. In the LP for a particular node i, the number of decision variables is equal to the number of outgoing edges from that node (i.e. $|O(i)|$), and the constraint matrix $\mathbf{A}'$ is a square matrix with length also equal to $|O(i)|$.

$|O(i)|$ of course varies for each node, but an upper limit is the total number of nodes $|\mathcal{N}|$. In the worst case, the simplex method will therefore require $\mathcal{O}(|\mathcal{N}|^2)$ computations to perform a single iteration. Because it is possible (though unlikely) for the simplex method to visit every vertex of the feasible set before finding the optimal solution, the maximum number of iterations required is approximately $2^{|\mathcal{N}|}$ [3]. Finally, with one LP for every node, there are a total of $|\mathcal{N}|$ LPs to be solved. The worst-case computational requirement is therefore $\mathcal{O}(2^{|\mathcal{N}|}|\mathcal{N}|^3)$ operations.

4.1.3 The 2-Node Case

For this discussion only, make the assumption that $\alpha_i = 1 \; \forall \; i \in \mathcal{N}$, in which case V_i represents the exact probability of Player 1 being ambushed when he starts at node i. Now suppose Player 1 is situated at node 0 and can move to either node 1 or node 2, with expected outcomes of V_1 and V_2 respectively. In this simple case, it is relatively straightforward to find a closed-form solution for each player's optimal strategy, and the result is useful for demonstrating that the optimal strategy for Player 1 is not necessarily intuitive.

If the game terminates with a successful ambush, then the game matrix at node 0 is:

$$\mathbf{A} = \begin{bmatrix} 1 & V_1 \\ V_2 & 1 \end{bmatrix} \tag{4.2}$$

Note that this game has no saddle-point in pure strategies since the minimax value is 1, but the maximin value is $\max(V_1, V_2)$. Therefore, we need to examine the class of mixed strategies to find the game solution.

Let Player 1's strategy be $\mathbf{p} = \begin{bmatrix} p & 1-p \end{bmatrix}^T$, so that he chooses node 1 with probability p and node 2 with probability $1-p$. Similarly, let Player 2's strategy be $\mathbf{q} = \begin{bmatrix} q & 1-q \end{bmatrix}^T$. The expected outcome of the game at node 0 is therefore:

$$\begin{aligned} V_0(p,q) &= \mathbf{p'Aq} \\ &= \begin{bmatrix} p & 1-p \end{bmatrix} \begin{bmatrix} 1 & V_1 \\ V_2 & 1 \end{bmatrix} \begin{bmatrix} q \\ 1-q \end{bmatrix} \\ &= pq + (1-p)(1-q) + p(1-q)V_1 + (1-p)qV_2 \end{aligned} \tag{4.3}$$

Let P and Q be the set of mixed strategies for Players 1 and 2 respectively. The optimal strategies are:

$$p^* = \arg\min_P \max_Q \{V_0(p,q)\} \tag{4.4}$$

$$q^* = \arg\max_Q \min_P \{V_0(p,q)\} \tag{4.5}$$

To find p^* and q^*, first calculate the partial derivatives of the value function:

$$\frac{\partial V_0}{\partial p} = (2 - V_1 - V_2)q - (1 - V_1) \tag{4.6}$$

$$\frac{\partial V_0}{\partial q} = (2 - V_1 - V_2)p - (1 - V_2) \tag{4.7}$$

Setting the partial derivatives to zero gives the following:

$$\frac{\partial V_0}{\partial q} = 0 \implies p = \hat{p} = \frac{1 - V_2}{2 - V_1 - V_2} \tag{4.8}$$

$$\frac{\partial V_0}{\partial p} = 0 \implies q = \hat{q} = \frac{1 - V_1}{2 - V_1 - V_2} \tag{4.9}$$

We will now show that $p^* = \hat{p}$ and $q^* = \hat{q}$. Since $0 \leq V_1 \leq 1$ and $0 \leq V_2 \leq 1$ (because $\alpha_i = 1 \; \forall \; i \in \mathcal{N}$), both $\hat{p}$ and $\hat{q}$ lie within $[0, 1]$. The value of the game with those strategies is:

$$\begin{aligned} V_0(\hat{p}, \hat{q}) &= \frac{(1 - V_1)(1 - V_2)}{(2 - V_1 - V_2)^2} + \frac{(1 - V_1)(1 - V_2)}{(2 - V_1 - V_2)^2} + \frac{(1 - V_2)^2 V_1}{(2 - V_1 - V_2)^2} + \frac{(1 - V_1)^2 V_2}{(2 - V_1 - V_2)^2} \\ &= \frac{V_1(1 - V_2)^2 + 2(1 - V_1)(1 - V_2) + V_2(1 - V_1)^2}{(2 - V_1 - V_2)^2} \\ &= \frac{1 - V_1 V_2}{2 - V_1 - V_2} \end{aligned}$$

From (4.6) we see that $\frac{\partial V_0}{\partial p}$ is a monotonically increasing function of q, which is negative for $q < \hat{q}$ and positive for $q > \hat{q}$. Since Player 1's objective is to minimize V_0, his optimal policy $\bar{p}(q)$ in response to Player 2's policy q is:

$$\bar{p}(q) = \begin{cases} 1 & , \quad q > q^* \\ 0 & , \quad q < q^* \end{cases}$$

(For $q = \hat{q}$, since $\frac{\partial V_0}{\partial p} = 0$, all values of p will lead to the same value of V_0.)

Finally, we need to show that $V_0(\bar{p}, q) < V_0(\hat{p}, \hat{q}) \; \forall \; q \neq \hat{q}$. For $q < \hat{q}$, we have from (4.3) and (4.9):

$$\begin{aligned} V_0(\bar{p}, q) &= V_0(1, q) \\ &= V_1 + (1 - V_1)q \\ &< V_1 + \frac{(1 - V_1)^2}{2 - V_1 - V_2} \\ &< \frac{1 - V_1 V_2}{2 - V_1 - V_2} \\ &< V_0(\hat{p}, \hat{q}) \end{aligned}$$

And for $q > \hat{q}$:

$$\begin{aligned} V_0(\bar{p}, q) &= V_0(0, q) \\ &= 1 - (1 - V_2)q \\ &< 1 - \frac{(1 - V_2)(1 - V_1)}{2 - V_1 - V_2} \\ &< \frac{1 - V_1 V_2}{2 - V_1 - V_2} \\ &< V_0(\hat{p}, \hat{q}) \end{aligned}$$

We have now shown that:

$$V_0(\bar{p}, q) < V_0(\hat{p}, \hat{q}) \; \forall \; q \neq \hat{q}$$

$$\Rightarrow \; q^* = \hat{q}$$

Similarly, it can be shown that:

$$V_0(p, \bar{q}) < V_0(\hat{p}, \hat{q}) \; \forall \; p \neq \hat{p}$$

$$\Rightarrow \; p^* = \hat{p}$$

And therefore:

$$p^* = \frac{1 - V_2}{2 - V_1 - V_2} \tag{4.10}$$

$$q^* = \frac{1 - V_1}{2 - V_1 - V_2} \tag{4.11}$$

The result for p^* shows that Player 1's optimal probability for moving to node 1 *increases* with V_1. In other words, Player 1's optimal mixed strategy is biased towards the more dangerous node, which may appear to be somewhat counter-intuitive initially.

The intuition behind this result is that Player 2 is also an intelligent agent and will pursue his own optimal strategy, which, as the result for q^* shows, is to place his ambush with a higher probability at the safer node. Therefore, although that node may be safer for Player 1 if he can arrive without being ambushed, he also has a higher probability of being ambushed along the way.

4.2 Game 2

This game is identical to Game 1 except now Player 2 can mount only one ambush over the course of the entire game, as opposed to one ambush at every stage of the game. This game may be a good model for a situation where the commander of the hostile forces receives periodic updates about Player 1's position, and must make the decision at some stage to commit all his forces to an ambush location.

If Player 2 has not yet committed to an ambush, then one stage of the game proceeds as follows:

1. Player 2 observes Player 1's position.

2. Player 1 chooses an adjacent node to move to.

3. Player 2 chooses an adjacent node at which to place an ambush, or does not commit and does nothing until the next stage.

In theory, Player 2 could choose a non-adjacent node at which to place an ambush, but he would be making his decision without the benefit of the useful information of Player 1's next move. Since there is no incentive to choose an ambush location earlier than necessary, it would never be in his interest to do so.

As in Game 1, a dynamic programming approach can be used here to determine the optimal strategy for Player 1. The game matrix at node i, where Player 1 has the option of moving to nodes $j_1 \dots j_n$, is:

$$\mathbf{A} = \begin{bmatrix} \alpha_{j_1} & 0 & \dots & 0 & V_{j_1} \\ 0 & \alpha_{j_2} & \dots & 0 & V_{j_2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \alpha_{j_n} & V_{j_n} \end{bmatrix} \tag{4.12}$$

The first n columns of the matrix correspond to Player 2's decision to stage an ambush at one of the adjacent nodes to Player 1's current position. The last column corresponds to Player 2's decision to not commit to an ambush location at this stage of the game, in which case the new expected outcome of the game is equal to the expected outcome at the node to which Player 1 moves.

The approach to solving this game is identical to the approach for Game 1, except for the change in the game matrix at each node, and it is detailed in Algorithm 4.2.

The complexity of this solution is also almost identical to the complexity of the solution for Game 1. The only difference is that the constraint matrix $\mathbf{A}'$ has one extra row, corresponding to Player 2's option to wait. However, the single extra constraint does not affect the order of the number of operations. In the worst case, the number of operations required is $\mathcal{O}(2^{|\mathcal{N}|}|\mathcal{N}|^3)$, as it is for Game 1.

Algorithm 4.2 Computing the optimal strategy for Game 2

```
/* U = set of unlabeled nodes */
U ← N \ n_d

/* L = set of labeled nodes */
L ← {n_d}

/* O(i) = set of nodes linked from node i */
O(i) = {j ∈ N | (i, j) ∈ E}

V_{n_d} = 0

while U ≠ ∅ do
  for all i ∈ U | O(i) ∈ L do
    Construct A from (4.12)
    Solve for V_i and p_i^* using (2.1)
    U ← U \ i
    L ← L ∪ {i}
  end for
end while
```

4.3 Game 3

In this game, Player 2 observes Player 1's location at every stage of the game. Just as in Game 2, Player 2 can mount only one ambush and therefore at each stage chooses whether to commit to an ambush location or wait and defer his decision. The difference is now that the outcome of a successful ambush is proportional to the amount of time in advance at which Player 2 committed to the ambush location.

This game corresponds to the situation where the hostile forces are faced with a tradeoff: early commitment to an ambush location gives them more time to prepare an ambush, and the resulting ambush (if successful) will be more effective. Later commitment to an ambush location means a less effective ambush, but more certainty that the convoy will appear at the ambush site.

4.3.1 Problem Formulation and Solution

The key difference between this game and Game 2 is the following: instead of an outcome of α_j for an ambush at node j, the outcome is now $t\alpha_j$ where t is the time

between Player 2's commitment to prepare an ambush at j and Player 1's arrival at j.

As in the previous two games, a dynamic programming approach can be used here to find the optimal strategy for Player 1, although forming the game matrix at a particular node is substantially more complicated than it is for Games 1 or 2.

At a given node i, Player 1 has the option to move to any adjacent node $j \mid (i,j) \in \mathcal{E}$. Player 2 has the following options:

1. Prepare an ambush at an adjacent node j
2. Prepare an ambush at a non-adjacent node k (between i and the destination)
3. Wait until the next turn

In the first case, we have:

$$A(j_1, j_2) = \begin{cases} t_{ij}\alpha_j & \text{if } j_1 = j_2 = j \\ 0 & \text{if } j_1 \neq j_2 \end{cases}$$

where t_{ij} is the time to travel directly from node i to node j.

In the third case, the game matrix entry is simply the expected value of the game starting from the node which Player 1 moves to:

$$A(j, \text{wait}) = V_j$$

The second case is more of a challenge. Suppose Player 1 is at node j and Player 2 has just committed to an ambush at node k. Assume that node k is not necessarily adjacent to node j, in which case there may be more than one possible route between the two nodes. Let $\mathcal{R}_{jk}$ be the set of possible routes from node j to node k, and let $\mathcal{P}^r_{jk}$ be the probability that Player 1 follows route $r \in \mathcal{R}_{jk}$ from j to k under his optimal strategy. Then the expected outcome of the game is given by:

$$E[V] = \left(\sum_{r \in \mathcal{R}_{jk}} \mathcal{P}^r_{jk} t^r_{jk} \right) \alpha_k \tag{4.13}$$

where t^r_{jk} is the time to traverse from node j to node k along route r.

Let y_{jk} be the expression in parentheses:

$$y_{jk} = \sum_{r \in \mathcal{R}_{jk}} \mathcal{P}^r_{jk} t^r_{jk} \tag{4.14}$$

Suppose Player 1 is at node i, and y_{jk} is already known for all adjacent nodes j and all subsequent nodes k. Now suppose Player 2 commits to an ambush at node k, while Player 1 chooses to move to adjacent node j. We can express the corresponding game matrix element in terms of y_{jk} as follows:

$$\begin{aligned} A(j,k) &= \left(\sum_{r \in \mathcal{R}_{jk}} \mathcal{P}^r_{jk} (t_{ij} + t^r_{jk}) \right) \alpha_k \\ &= \left(\sum_{r \in \mathcal{R}_{jk}} \mathcal{P}^r_{jk} t^r_{jk} + \left(\sum_{r \in \mathcal{R}_{jk}} \mathcal{P}^r_{jk} \right) t_{ij} \right) \alpha_k \\ &= (y_{jk} + p_{jk} t_{ij}) \alpha_k \end{aligned} \tag{4.15}$$

where p_{jk} is the overall probability under Player 1's optimal strategy of traveling to node k when he starts at node j.

We have now expressed $A(j,k)$ for node i in terms of quantities defined at other nodes between i and the target, which strongly suggests that a dynamic programming approach can be used to find the optimal strategy. The game matrix at node i is:

$$\mathbf{A} = \left[\begin{array}{c|c|c} \mathbf{A}_1 & \mathbf{A}_2 & \mathbf{A}_3 \end{array} \right] \tag{4.16}$$

where:

$$\mathbf{A}_1 = \begin{bmatrix} t_{ij_1}\alpha_{j_1} & 0 & \dots & 0 \\ 0 & t_{ij_2}\alpha_{j_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & t_{ij_n}\alpha_{j_n} \end{bmatrix}$$

$$\mathbf{A}_2 = \begin{bmatrix} (y_{j_1k_1} + p_{j_1k_1}t_{ij_1})\alpha_{k_1} & (y_{j_1k_2} + p_{j_1k_2}t_{ij_1})\alpha_{k_2} & \dots & (y_{j_1k_m} + p_{j_1k_m}t_{ij_1})\alpha_{k_m} \\ (y_{j_2k_1} + p_{j_2k_1}t_{ij_2})\alpha_{k_1} & (y_{j_2k_2} + p_{j_2k_2}t_{ij_2})\alpha_{k_2} & \dots & (y_{j_2k_m} + p_{j_2k_m}t_{ij_2})\alpha_{k_m} \\ \vdots & \vdots & & \vdots \\ (y_{j_nk_1} + p_{j_nk_1}t_{ij_n})\alpha_{k_1} & (y_{j_nk_2} + p_{j_nk_2}t_{ij_n})\alpha_{k_2} & \dots & (y_{j_nk_m} + p_{j_nk_m}t_{ij_n})\alpha_{k_m} \end{bmatrix}$$

$$\mathbf{A}_3 = \begin{bmatrix} V_{j_1} \\ \vdots \\ V_{j_n} \end{bmatrix}$$

Because the game matrix contains one column for every node between the current node and the destination, it can become quite large and the corresponding linear program may require a considerable amount of computational effort to solve. However, this process can be streamlined by first removing any *dominated columns* from $\mathbf{A}$. One column is said to be dominated by another if each element of the first column is less than or equal to the corresponding element in the second column, with at least one element strictly less. In that case, Player 2 will never have reason to choose the first column, and therefore it can be excised from the matrix before solving for the optimal strategies.

Having determined the game matrix, all that remains is to determine y_{ik} and p_{ik} for all nodes k between the current node i and the destination, given y_{jk} and p_{jk} for all adjacent nodes j. These values can be found by essentially taking a probability-weighted combination of the values at the adjacent nodes:

$$y_{ik} = \sum_j p_{ij}(y_{jk} + p_{jk}t_{ij}) \tag{4.17}$$

$$p_{ik} = \sum_j p_{ij}p_{jk} \tag{4.18}$$

We now have all the elements needed to develop an algorithm for finding the optimal strategy for Player 1 in Game 3. This algorithm is shown in Algorithm

4.3, and works as follows: at the destination node n_d, set $V_{n_d} = 0$. Then, working backwards from the destination, at each node i construct $\mathbf{A}$, solve the associated LP, and calculate y_{ik} and p_{ik} for all nodes k between i and the destination. At the conclusion of this process, the optimal probability p_{ij} will be known for every pair of adjacent nodes i and j. For non-adjacent nodes i and k, p_{ik} will specify the total probability of reaching k from i using any feasible route.

Algorithm 4.3 Computing the optimal strategy for Game 3

```
/* U = set of unlabeled nodes */
U ← N \ n_d

/* L = set of labeled nodes */
L ← {n_d}

/* O(i) = set of nodes linked from node i */
O(i) = {j ∈ N | (i,j) ∈ E}

V_{n_d} = 0
p_ij ← 0 ∀ i,j ∈ N
y_ij ← 0 ∀ i,j ∈ N

while U ≠ ∅ do
  for all i ∈ U | O(i) ∈ L do
    Construct A from (4.16)
    Solve for V_i and p_ij ∀ j ∈ O(i) using (2.1)
    for k ∈ L \ O(i) do
      for j ∈ O(i) do
        p_ik ← p_ik + p_ij * p_jk
        y_ik ← y_ik + p_ij * (y_jk + p_jk * t_ij)
      end for
    end for
    U ← U \ i
    L ← L ∪ {i}
  end for
end while
```

4.3.2 Complexity

As in Games 1 and 2, for Game 3 an LP must be solved at each node. The number of decision variables at a node i is again equal to $|O(i)|$, the number of outgoing edges

from that node. However, the number of rows of the constraint matrix $\mathbf{A}$ is now equal to one plus the number of nodes between i and the destination. Although this number will typically be much greater than $|O(i)|$, particularly for nodes further away from the destination, it has the same upper limit of approximately $|\mathcal{N}|$ and therefore the order of the number of operations for solving the LPs is once again $\mathcal{O}(2^{|\mathcal{N}|}|\mathcal{N}|^3)$, as it is for Games 1 and 2.

In addition to solving the LPs, however, the solution to Game 3 also requires the calculation of p_{ik} an y_{ik} for every non-adjacent node k between the current node i and the destination. From (4.17) and (4.18), the number of multiplications required is $2|O(i)|$ for y_{ik} and $|O(i)|$ for p_{ik}, or $\mathcal{O}(|\mathcal{N}|)$ in total. The upper limit for the number of non-adjacent nodes k is $|\mathcal{N}|$, and these calculations must be carried out for each node, for a total of $|\mathcal{N}|$ times. Therefore, the total number of operations required for solving Game 3 is $\mathcal{O}(2^{|\mathcal{N}|}|\mathcal{N}|^3 + |\mathcal{N}|^3)$ or simply $\mathcal{O}(2^{|\mathcal{N}|}|\mathcal{N}|^3)$.

Although the order is the same as for Games 1 and 2, in practice the actual number of operations may be significantly higher, because of the increased number of constraints and the extra calculations for determining y_{ik} and p_{ik}.

4.4 Example Applications

To illustrate some of the ideas described in this chapter, optimal strategies were computed for a some example scenarios. For reference, a summary of the various multistage ambush games can be found in Table 4.1. As in Section 3.3, all computations were carried out on a Pentium 4 2.20 GHz desktop PC with 512 MB of RAM, running Windows XP.

4.4.1 3x3 Grid

The optimal strategies for the three multi-stage ambush games were found for a scenario taking place on the 3x3 grid (plus origin and destination nodes) shown in Figure 4-1. The origin node is numbered 1 and the destination node is numbered 11. Each column of the grid represents a "stage" of the game, and at each stage Player

Game	Scenario
1a	1 ambush per stage Game ends with a successful ambush
1b	1 ambush per stage Game continues after a successful ambush
2	1 ambush per game
3	1 ambush per game Outcome proportional to ambush preparation time

Table 4.1: Summary of multistage games

1 can move one unit in the horizontal direction and up to 1 unit up or down in the vertical direction.

To simplify the results, α was taken to be 1 for each node (except the origin and destination, for which $\alpha = 0$). The time to traverse each edge was also taken to be 1. Although the resulting scenario is very simple and may not represent a real-life situation, it is nevertheless very useful for illustrating the differences between the various multistage games.

The results are shown in Table 4.2. The computation times to determine the solutions for Games 1 and 2 are approximately equal, which is to be expected since they all use essentially the same algorithm. For Game 3, the computational requirements are higher by a significant amount, due to the larger game matrices at each node, and the extra calculations to determine y_{ik} and p_{ik} values.

An expected outcome of 0.795 for Game 1a implies that Player 1 has exactly a 79.5% chance of being ambushed if he pursues his optimal strategy. The corresponding value of 1.204 for Game 1b is, as expected, higher because it reflects the scenario in which Player 1 can be ambushed multiple times on the way to the destination. On the other hand, the outcome of 0.467 for Game 2 implies that Player 1 has only a 46.7% chance of being ambushed when Player 2 is constrained to choosing only one ambush location for the entire game. Finally, the expected outcome of 1.5 for Game 3 indicates that the expected outcome of the game is for Player 1 to be ambushed with Player 2 committing to the ambush 1.5 stages before Player 1's arrival at the chosen ambush site.

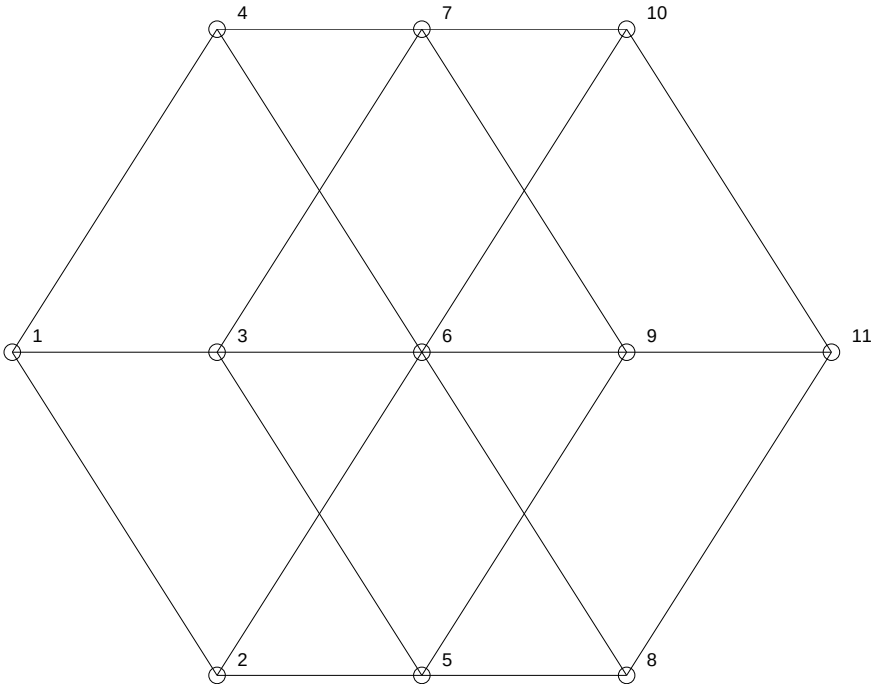

Figure 4-1: Example graph

Game	**Computation Time**	**Expected Outcome**
1a	0.078 s	0.795
1b	0.104 s	1.204
2	0.073 s	0.467
3	0.354 s	1.500

Table 4.2: Results for multistage games on 3x3 grid

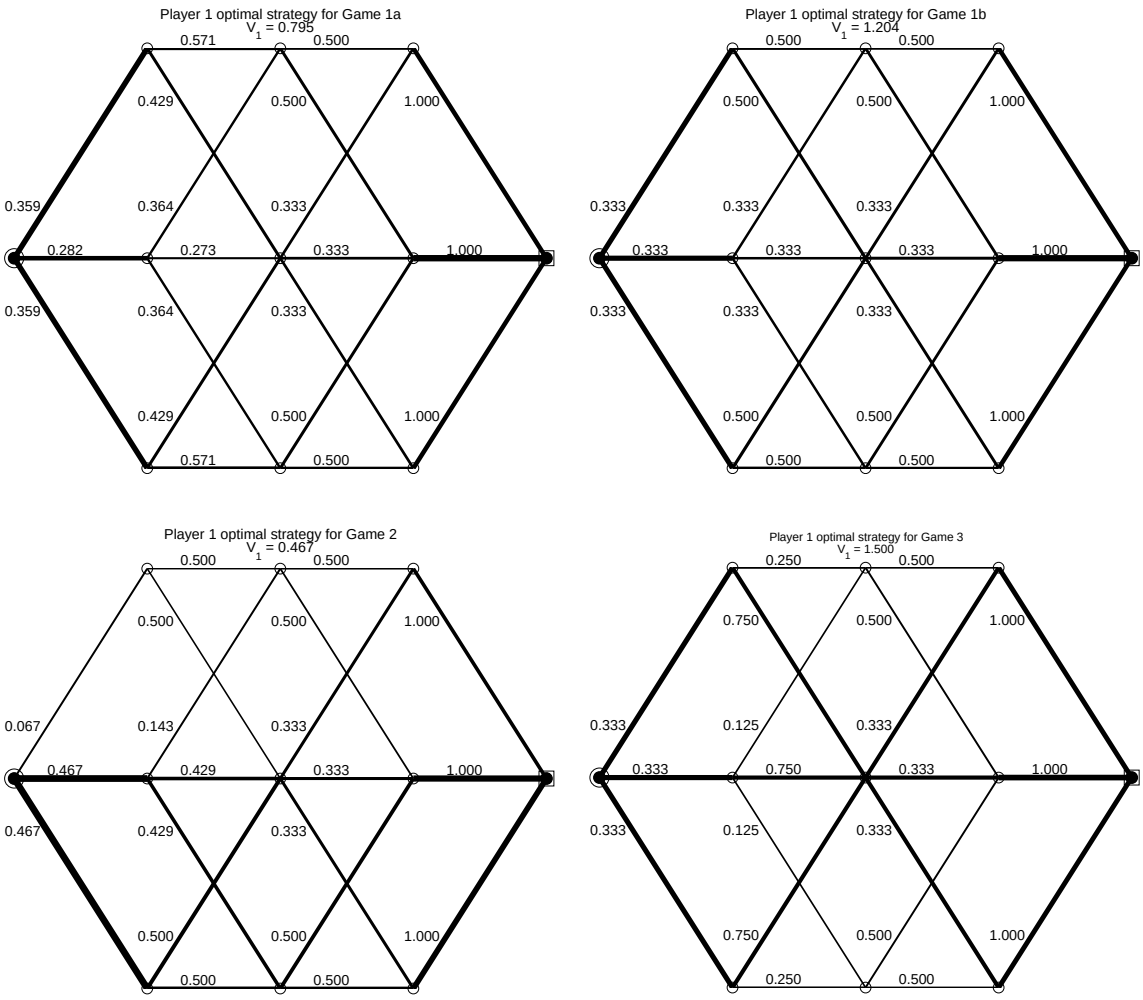

Figure 4-2: Optimal strategies for multi-stage ambush games on 3x3 grid

The optimal strategies for these four scenarios are shown in Figure 4-2. The edge labels represent the transitional probabilities of Player 1 traversing between the corresponding nodes. That is, p_{ij} for the edge connecting nodes i and j is the probability that Player 1, under his optimal strategy, will travel to node j given that he is already at node i.

4.4.2 Cambridge Scenario

Game 3 was solved for the Cambridge scenario described in Section 3.3. One necessary change made to the scenario was to remove certain edges so that all edges would be directed towards the origin, allowing Algorithm 4.3 to be applied. The time taken

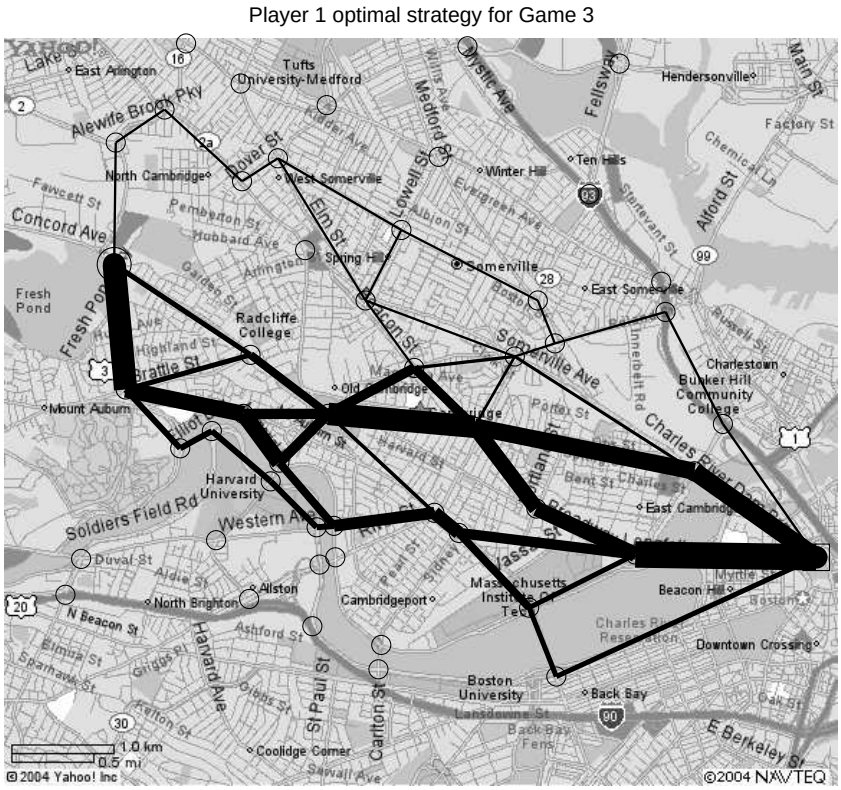

Figure 4-3: Optimal strategy for Cambridge scenario, Game 3.
Darker lines correspond to higher probability edges.

to travel between two nodes was assumed to be proportional to the distance between the nodes.

The optimal strategy for Player 1 is shown in Figure 4-3. The computation time was 1.22 s. Interestingly, the optimal strategy is heavily biased towards the more direct routes to the target, which may reflect the fact that a more indirect route allows Player 2 more time to prepare an ambush.

Chapter 5

Conclusion

5.1 Summary

The major contribution of this thesis has been the development of efficient methods for computing the optimal routing strategies to avoid ambush in a number of ambush games. Chapter 3 looked at single-stage ambush games, in which Player 2 chooses his ambush locations at the same time as Player 1 chooses his route. A network flow formulation of this problem eliminates the need to explicitly determine all routes from origin to destination, and also leads to a linear program that can be solved rapidly with inexpensive computer hardware, even for relatively complex scenarios.

In Chapter 4, the focus shifted to multi-stage ambush games, in which Player 2 observes Player 1's position en route to the destination and uses this information to select ambush locations. Three different games were defined: one in which Player 2 can place an ambush at every stage of the game, one in which Player 2 can place only one ambush for the entire game, and one in which Player 2 can place only one ambush and his payoff is proportional to the ambush preparation time.

For these games, a dynamic programming approach was used to determine the optimal mixed strategy for Player 1. Again, this approach avoids the need to compile a list of routes from origin to destination, and the solution methods are computationally efficient enough to be applied in a real-life setting.

The solutions to these ambush games can be applied to a wide range of realistic

ambush situations. In particular, the problem of transporting a VIP through a hostile area on a regular schedule calls for random routing, and these methods can be used "in the field" to efficiently and rigorously compute the optimal randomization.

5.2 Future Work

For future work, it may be advisable to revisit some of the assumptions made in the problem formulation. The assumption that all ambushes occur at nodes (street intersections) rather than along edges (between intersections) has already been discussed, and may be dropped in situations where it is not valid. Another key assumption is that Player 2 has full information about the scenario (e.g. origin and destination nodes) and is intelligent enough to determine his own optimal strategy. If this assumption is too generous to Player 2, then Player 1 may be able to exploit this fact and obtain a better outcome with different strategies from those proposed in this thesis.

Another area of work would be to compute optimal strategies for variants of the games described here. One realistic example is the single-stage game in which one driver must pick up a number of VIPs from different locations and drop them off at different locations. (The solution to that game may also be applicable to an armored delivery van that must stop at a number of different banks.) Another example is the multi-stage Game 3 in which Player 2 has the capability to mount several ambushes, but the payoff is still proportional to the time in advance at which he made the commitment.

Finally, the solution approach to the multi-stage games has an inherent limitation: it requires all edges to be directed from the origin to the destination, to avoid having to solve a simultaneous system of optimization problems. To address this limitation, Player 1's state during the game can be redefined as a combination of his location node and the current game stage. The resulting state space will increase in size due to the addition of time as a state variable, but it will allow the dynamic programming approach to handle completely arbitrary graphs.

Bibliography

[1] T. Basar and G.J. Olsder. *Dynamic Noncooperative Game Theory.* SIAM, Philadelphia, PA, 2nd edition, 1999.

[2] V.J. Baston and F.A. Bostock. A continuous game of ambush. *Naval Research Logistics*, 34:645–654, 1987.

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

[4] A.Y. Garnaev. On a Ruckle problem in discrete games of ambush. *Naval Research Logistics*, 44:353–364, 1998.

[5] S. Gentry. *Dancing cheek to cheek: Haptic communication between partner dancers and swing as a finite state machine.* PhD thesis, Massachusetts Institute of Technology, 2005.

[6] S. Gentry and E. Feron. Modeling musically meaningful choreography. In *IEEE Systems, Man, Cybernetics Conference*, 2004.

[7] J.P. Hespanha, Y.S. Ateskan, and H.H. Kizilocak. Deception in non-cooperative games with partial information. In *2nd DARPA-JFACC Symposium on Advances in Enterprise Control*, July 2000.

[8] G. Owen. *Game Theory.* AP, San Diego, CA, 3rd edition, 1995.

[9] P. Root, J. De Mot, and E. Feron. Randomized path planning with deceptive strategies. In *American Control Conference*, June 2005.

[10] P.J. Root. Collaborative UAV path planning with deceptive strategies. Master's thesis, Massachusetts Institute of Technology, 2005.

[11] W.H. Ruckle. Ambushing random walks II: Continuous models. *Operations Research*, 29(1):108–120, 1981.

[12] W.H. Ruckle. *Geometric games and their applications*. Pitman, Boston, MA, 1983.

[13] W.H. Ruckle, R. Fennell, P.T. Holmes, and C. Fennemore. Ambushing random walks I: Finite models. *Operations Research*, 24(2):314–324, 1976.

[14] W.H. Ruckle and J.R. Reay. Ambushing random walks III: More continuous models. *Operations Research*, 29(2):121–129, 1981.

[15] I.D. Woodward. Discretization of the continuous ambush game. *Naval Research Logistics*, 50:515–529, 2003.

[16] N. Zoroa, P. Zoroa, and J. Fernandez-Saez. New results on a Ruckle problem in discrete games of ambush. *Naval Research Logistics*, 48:98–106, 2001.