Your Title Goes Here (It Can Be Really Really Really Really Long)

Your Name Here

Abstract Motivated by the goal recognition (GR) and goal recognition design (GRD) problems in the artificial intelligence (AI) planning domain, we introduce and study two natural variants of the GR and GRD problems with strategic agents, respectively. More specifically, we consider game-theoretic (GT) scenarios where a malicious adversary is aiming to cause damage to some target in an (physical or virtual) environment monitored by a defender. The adversary is interested in attacking a single target and must take a sequence of actions in order to attack the target. In the GTGR and GTGRD settings, the defender's goal is to identify the adversary's intended target from observing the adversary's actions so that he/she can strengthens the target's defense against the attack. In addition, in the GTGRD setting, the defender can alter the environment (e.g., adding roadblocks) in order to better distinguish the goal/target of the adversary.

We propose to model GTGR and GRGRD settings as zero-sum stochastic games with incomplete information about the adversary's intended target. The games are played on graphs where vertices are states and edges are adversary's actions. For the GTGR setting, we show that if the defender is restricted to playing only stationary strategies, the problem of computing optimal strategies (for both defender and adversary) can be formulated and represented compactly as a linear program. For the GTGRD setting, where the defender can choose K edges to block at the start of the game, we formulate the problem of computing optimal strategies as a mixed integer program, and present a heuristic algorithm based on LP duality and greedy methods. Experiments show that our heuristic algorithm achieves good performance (i.e., close to defender's optimal value) with better scalability compared to the mixed-integer programming approach.

In contrast with our research, existing work, especially on GRD problems, have focused almost exclusively on decision-theoretic paradigms, where the adversary chooses its actions without taking into account the fact that they may be observed by the defender. As such an assumption is unrealistic in GT scenarios, our proposed models and algorithms fill a significant gap in the literature.

A nice abstract goes here.

Acknowledgments

Some acknowledgments go here.

Your Title Goes Here (It Can Be Really Really Really Really Long)

Your Name Here

A departmental senior thesis submitted to the
Department of Computer Science at Trinity University
in partial fulfillment of the requirements for graduation
with departmental honors.

April 1, 2005

Thesis Advisor

Department Chair

Associate Vice President
for
Academic Affairs

Distribution options for digital thesis:

[X] Open Access (full-text discoverable via search engines)

[ ] Restricted to campus viewing only (allow access only on the Trinity University campus via digitalcommons.trinity.edu)

Your Title Goes Here (It Can Be Really Really Really Really Long)

Your Name Here

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Researchers in both artificial intelligence (AI) and psychology share an interest in developing techniques for discerning the intentions of agents through observation [23, 7]. AI researchers have referred to these types problems as *goal recognition problems* (GR) or, more generally, *plan recognition problems* [25]. Plan and goal recognition models have many applications, both for cooperative and adversarial relationships between the agent in question and the party attempting to recognize their goal. Goal recognition has been used in designing software for personal assistants [16, 17, 18]; for robots that interact with humans in cooperative work settings such as homes, offices, and hospitals [26, 8]; for intelligent tutoring systems that can recognize sources of confusion for a student based on their interactions with the system [14, 6, 12, 15]; and for security applications such as recognizing the goal of terrorists [5].

Goal recognition research focuses primarily on developing more effective, and more efficient techniques for recognizing the goal of an agent through observing the agent's prior actions. Figure 1.1 illustrates a scenario in which a potentially malicious agent attempts to reach one of three goals. The agent in cell $E3$ can move to any adjacent cell on the

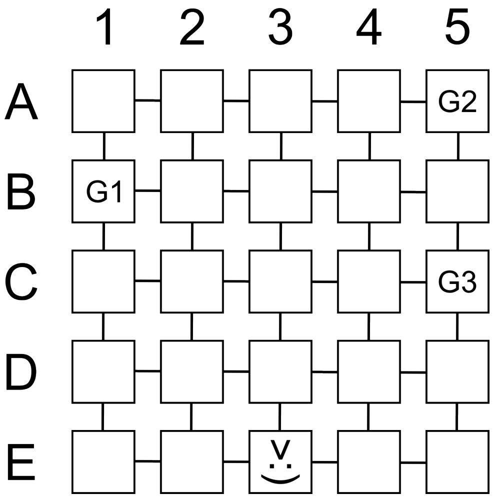

Figure 1.1: An Example Goal Recognition Problem.

graph. Cells $A5, B1$, and $C5$, hold potential goal states for the agent. While the agent has a particular goal they want to reach, the observer is only aware of what goals the agent could potentially be working towards. In this scenario, a GR model could help an onlooker discern which goal the agent intends to reach.

Existing research into GR models has focused on partially strategic, or non strategic agents. While agents attempt to reach their secret goal at minimum cost, they do not explicitly reason about their interaction with observers. When an observer's recognition of the agent's goal affects the agent in some way, then it is in the agent's best interest to be *fully strategic*, and to consider how their actions might affects the observer's recognition. As a result, the observer will need to take the agent's strategic reasoning into account when making decisions.

1.0.1 Game-Theoretic Goal Recognition Problems in Security Domains

Goal Recognition settings with strategic agents can model many real-world (physical and cyber) security scenarios between an adversary and a defender. In physical security domains, the adversary must make a sequence of physical movements to reach their target. In cyber security domains, this could be achieving a sequence of actions for necessary subgoals when carrying out an attack. In any case, the defender can benefit from recognizing the adversary's intended target in advance. Moving forward, we will take a game-theoretic approach to modeling such problems.

Consider the security scenario in Figure 1.1, where an agent (i.e., a notorious art thief) wants to reach its intended target and carry out an some devilish action. Meanwhile we the observer must try to recognize the agent's goal as early as possible. Suppose that once we recognize the agent's goal, we can strengthen the agent's target to defend against the attack. The more time we have between recognition and the actual attack, the less successful the attack will be. In this scenario, the agent can not simply take the shortest path to its goal, since their intentions could quickly become clear to the observer. At the same time, the agent should try to reach its goal in a reasonably short amount of time, as a very long path could allow the observer time to strengthen all the targets. An optimal agent would need to explicitly reason about the tradeoffs between the cost of its path (e.g., path length) and the cost of an early discovery.

1.0.2 Game-Theoretic Goal Recognition Design Problems in Security Domains

So far we have discussed the defender's task in recognizing goals. However, the task could become extremely difficult in general. For instance, going back to our security example in

Figure 1.1, if the agent moves up to $D3$, the observer cannot make any informed deductions. Unfortunately, if the agent moves along any one of the shortest paths to goal $G3$, then throughout its entire path, we cannot deduce whether its goal is either $G2$ or $G3$! Keren *et al.* introduced the concept of *worst-case distinctiveness* [9], the number of moves an agent can make before giving away their intentions. This illustrates one of the challenges observers can face when tackling GR scenarios. There can exist large sequences of ambiguous observations which prevent the observer from picking out the correct target .

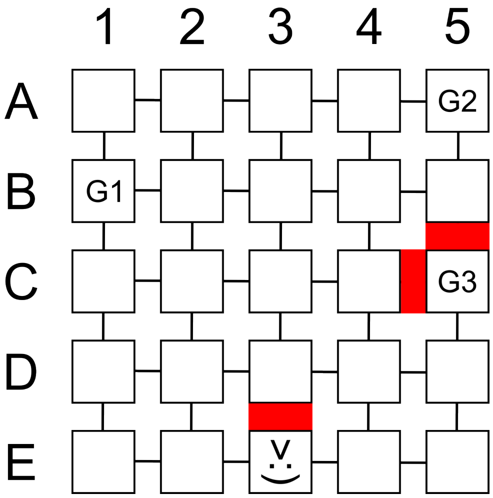

Figure 1.2: An Example Goal Recognition Design Problem with Blocked Edges.

The work of [9, 10] proposed an orthogonal approach to *modify the underlying environment of the agent*, in such a way that *the agent is forced to reveal its goal as early as possible*. They call this problem the goal recognition design (GRD) problem. For example, if we block the actions $(E3, up), (C4, right)$, $(C5, up)$ in our example problem, where we use tuples (s, a) to denote that action a is blocked from cell s, then the agent can make at most 2 actions (i.e., right to E4 then up to D4) before its goal is conclusively revealed. Figure 1.2 shows the blocked actions for the previous example.

In addition to studying the Game Theoretic Goal Recognition problem (GTGR), we will also explore methods of solving the Game-Theoretic Goal Recognition Design (GT-GRD) problem, where the observer can modify the underlying environment (i.e., adding K roadblocks) in order to improve their odds of determining the agent's target.

1.0.3 Related Work

GR and its more general forms, plan recognition and intent recognition, have been extensively studied [25] since their inception almost 40 years ago [23]. Researchers have made significant progress within the last decade through synergistic integrations of techniques ranging from natural language processing [27, 3] to classical planning [20, 21, 22] and deep learning [15]. The closest body of work to ours is the one that uses game-theoretic formulations, including an adversarial plan recognition model that is defined as an imperfect information two-player zero-sum game in extensive form [13], a model where the game is over attack graphs [1], and an extension that allows for stochastic action outcomes [4].

GR has a long history and extensive literature, but the field of GRD is relatively new. Keren *et al.* introduced the problem in their seminal paper [9], where they proposed a decision-theoretic STRIPS-based formulation of the problem. In the original GRD problem, the authors make several simplifying assumptions: (1) the observed agent is assumed to execute an optimal (i.e., cost-minimal) plan to its goal; (2) the actions of the agent are deterministic; and (3) the actions of the agent are fully observable. Since then, these assumptions have been independently relaxed, where agents can now execute boundedly-suboptimal plans [10], actions of the agents can be stochastic [28], and actions of the agents can be only partially observable [11]. Further, aside from all the decision-theoretic approaches above, researchers have also modeled and solved the original GRD problem using answer set programming [24]. The key difference between these works and ours is that ours introduced a game-theoretic formulation that can more accurately capture interactions between the agent and the observer in security applications.

1.0.4 Contributions

As a result of the strategic interaction in the GTGR and GTGRD scenarios, the cost-minimal plan (the solution concept in GR problem) and worst-case distinctiveness (the solution concept in GRD problem) are no longer suitable solution concepts since they do not reflect the behavior of strategic agents. Instead, our objective here is to formulate game-theoretic models of the agent's and observer's interactions under GR and GRD settings. More specifically, we will attempt to model GTGR and GRGRD settings as zero-sum stochastic games with incomplete information where the adversary's target is unknown to the observer. For the GTGR setting, we show that if the defender is restricted to playing only stationary strategies, the problem of computing optimal strategies (for both defender and adversary) can be formulated and represented compactly as a linear program. We will also explore approaches to solving a partially observable variant of the GTGR setting in which the observer can only know the position of the adversary for certain states. We will perform experiments to determine if the observer can improve their performance with a non-stationary strategy. For the GTGRD setting, where the defender can choose K edges to block at the start of the game, we formulate the problem of computing optimal strategies as a mixed integer program, and present a heuristic algorithm based on LP duality and greedy methods. We perform experiments to show that our heuristic algorithm achieves good performance (i.e., close to defender's optimal value) with better scalability compared to the mixed-integer programming approach.

1.1 Preliminary: stochastic games

In our two-player zero-sum single-controller stochastic game G, (a) we have a finite set S of states, and an initial state $s_0 \in S$, (b) given a state $s \in S$, a finite action set J_s and $I = I_s$ for the first player and for the second player, respectively, (c) given a state $s \in S$ and $j \in J_s$, a single-controller transition function $\chi(s, j)$ that deterministically maps the current state and action to a new state, and (d) given a state $s \in S$, $j \in J_s$, and $i \in I$, a reward function $r(s, i, j, \theta) \in \mathbb{R}$. Since this is a zero-sum game, without loss of generality, we define r to be the reward for player 2 and the reward of player 1 is the negative reward of player 2. We consider two-player zero-sum single-controller stochastic game where player 2 has incomplete information. In particular, the game consists of a collection of zero-sum single-controller stochastic games $\{G_\theta\}_{\theta \in B}$ and a probability distribution $P \in \Delta(B)$ over B. For our setting, we assume that each stochastic game G_θ could have different reward function r^θ, but all of the games $G_\theta' s$ have the same sets of states, actions, and transition rules. The game is played in stages over some finite time. First, a game G_θ is drawn according to P. The first player is informed of θ while the second player does not know θ. At each stage of game t with current state $s_t \in S$, the first player selects $j_t \in J_s$ and the second player selects $i_t \in I$, and s_{t+1} is reached according to $\chi(s_t, j_t)$. However, we assume that player 1 does not know o_t, and both of the players do not know $r^\theta(s_t, i_t, j_t)$. Note that player 2 can infer the action of player 1 given the new state since our transition function is deterministic. Hence, player 2 knows j_t, i_t, and s_{t+1}

The strategies of the players can be based on their own history of the previous states and strategies. In addition, player 1 can condition his strategies based on θ. We consider finite timestep at most T. Let $h_t^1 = (s_0, j_0, s_1, j_1, ..., j_{t-1}, s_t)$ and $h_t^2 = (s_0, j_0, i_0, s_1, ..., j_{t-1}, i_{t-1}, s_t)$ to denote a possible history of length t of player 1 and player 2 where $j_k \in J_{s_k}$ and $i_k \in I$

for $k = 1, ..., t$. Let $H^1_{s_t}$ and $H^2_{s_t}$ be the set of all possible histories of length t ended up at state s_t. Then, the sets of deterministic strategies for player 1 and player 2 are therefore $\prod_{t=0 \leq T, s_t \in S, h^1_{s_t} \in H^1_{s_t}} J_{s_t}$ and $\prod_{t=0 \leq T, s_t \in S, h^2_{s_t} \in H^2_{s_t}} I$, respectively. Indeed, for each possible history, the players need to select some actions. Naturally, the players mixed strategies are distributions over the deterministic strategies.

Definition 1.1.1. Given $\theta \in B$, $0 \leq t \leq T$, $s_t \in S$, $h^1_{s_t} \in H^1_{s_t}$, player 1's behavioral strategy $\sigma_1(\theta, h^1_{s_t}, j_{s_t})$ returns the probability of playing $j_{s_t} \in J_{s_t}$ such that $\sum_{j_{s_t} \in J_{s_t}} \sigma_1(\theta, h^1_{s_t}, j_{s_t}) = 1$. (Player 2's behavioral strategy σ_2 is defined similarly and does not depend on θ).

Definition 1.1.2. A behavioral strategy σ is stationary if and only if it is independent of any timestep t and depends only on the current state (i.e., $\sigma_1(\theta, h^1_s, j_s) = \sigma_1(\theta, \bar{h}^1_s, j_s)$ such that h^1_s and $\bar{h}^1_s$ have the same last state and σ_2 can be defined similarly).

Given a sequence $\{(s_t, i_t, j_t)\}_{t=1}^T$ of actions and states, the total reward for player 2 is $r_T = \sum_{t=1}^T r^\theta(s_t, i_t, j_t)$. Thus, the expected reward $\gamma_T(P, s_0, \sigma_1, \sigma_2) = \mathbf{E}_{P,s_0,\sigma_1,\sigma_2}[r_T]$ is the expectation of r_T over the set of stochastic games $\{G_\theta\}_{\theta \in B}$ given the the fixed initial state s_0 under P, σ_1, and σ_2, respectively.

Definition 1.1.3. The behavioral strategy σ_2 is a best response to σ_1 if and only if for all σ'_2, $\gamma_T(P, s_0, \sigma_1, \sigma_2) \geq \gamma_T(P, s_0, \sigma_1, \sigma'_2)$. The behavioral strategy σ_1 is a best response to σ_2 if and only if for all σ'_1, $\gamma_T(P, s_0, \sigma_1, \sigma_2) \leq \gamma_T(P, s_0, \sigma'_1, \sigma_2)$.

For two-player zero-sum games, the standard solution concept is the max-min solution: $\max_{\sigma_2} \min_{\sigma_1} \gamma_T(P, s_0, \sigma_1, \sigma_2)$. One can also define min-max solution $\min_{\sigma_1} \max_{\sigma_2} \gamma_T(P, s_0, \sigma_1, \sigma_2)$. For zero-sum games, the max-min value, min-max value, and Nash equilibrium values all coincide [2]. For simultaneous-move games this can usually be solved by formulating a linear program. In this work, we will be focusing on computing the max-min solution.

Chapter 2

Game Model

This chapter will introduce both the GTGR and GRGRD models.

2.0.1 Game-theoretic goal recognition model

Consider a deterministic environment such as the one in the introduction. We can model the environment with a graph in which the nodes correspond to the states and the edges connect neighboring states. Given the environment and the graph, as in many standard GR problems, the agent wants to plan out a sequence of moves (i.e., determining a path) to reach its target location of the graph. The target location is unknown to the observer, and the observer's goals are to identify the target location based on the observed sequence of moves and to make preventive measure to protect the target location.

We model this scenario as a two-player zero-sum game, between the agent/ adversary and the observer. Given the graph $G = (L, E)$ of the environment, the adversary is interested in a set of potential targets $B \subseteq L$ and has a starting position $s_0 \in L \setminus B$. The adversary's aim is to attack a specific target $\theta \in B$, which is chosen at random according to some prior probability distribution P. The observer does not know the target θ, and only the adversary knows its target θ. However, the observer knows the set of possible targets B and the adversary's starting position s_0. For any $s \in L$, we let $\nu(s)$ is the set of neighbors of s in the graph G.

The game is sequential and is played over several time-steps where both of the players move simultaneously. At each time-step, the observer selects a potential target in B to protect, and the agent moves from its current position to a neighboring node. With each time-step, the adversary and the observer will lose and gain a value d, respectively. In addition, if the observer protects the correct target location θ, an additional value of q will be added to the observer and subtracted from the adversary. The value d allows us to incentivize quick play for the adversary, but can be set to 0 such a penalty is not needed. Note that the value of d should never be negative, since that would allow the adversary to wander the board endlessly, increasing their score to infinity. The game ends when the attacker reaches its target θ, a value of u^θ will be added to the adversary's overall score, and u^θ will be subtracted from the observer's overall score. Notice that during the play of the game, the adversary does not observe the observer's action(s), and the players do not know of their current scores.

Because the adversary's moves may be stochastic in nature, and because the observer does not know the adversary's intended target, our setting is most naturally modeled as a *stochastic game with incomplete information* as defined in Section 1.1. More specifically, the set of states is L with an initial state s_0. Given a state $s \in S$, $\nu(s)$ is the action set for the adversary and B is the action set for the observer. Given a state $s \in S$ and $j \in \nu(s)$, the single-controller transition function $\chi(s, j) = j$. Indeed, the transition between states are controlled by the adversary only and is deterministic: From state s, where $s \neq \theta$, given attacker action $j \in \nu(s)$, the next state is j. The state θ is terminal: Once reached, the game ends. Given a state $s \in S$, $j \in \nu(s)$, and $i \in B$, we define the reward function

$r^{\theta}(s,i,j) \equiv r(s,i,j,\theta)$ from the observer's point of view as

$$r(s,i,j,\theta) = \begin{cases} d & j \neq \theta \; \& \; i \neq \theta \\ d+q & j \neq \theta \; \& \; i = \theta \\ d - u^{\theta} & j = \theta \; \& \; i \neq \theta \\ d+q-u^{\theta} & j = \theta \; \& \; i = \theta. \end{cases} \tag{2.1}$$

While, in theory, the game could go on forever if the adversary never reaches his target θ, because of the per-timestep cost of d, any path of sufficient length for the adversary would be dominated by the strategy of taking the shortest path to θ. Eliminating these dominated strategies allows us to set a finite bound for the duration of the game. Even in games where the value of d is set to 0, the defender could potentially play a uniformly random strategy that imposes a cost of $\frac{q}{|B|}$ per timestep. Therefore, an adversary strategy taking forever would achieve a value of $-\infty$ against the uniformly random defender strategy. In any Nash equilibrium the attacker will always reach their target in finite time.

We call this the game-theoretic goal recognition (GTGR) model. All of the definitions in Section 1.1 follow immediately from this game.

2.0.2 Game-theoretic goal recognition design model

As mentioned in the introduction, we also consider the game-theoretic goal recognition design (GTGRD) model. Formally, before the game starts, we allow the observer to block a subset of at most K actions from the game. In our model, that corresponds to blocking at most K edges from the graph. In our game, blocking an edge does not prevent the adversary from taking the action, but the adversary would incur a cost by taking that action. After placing the blocks, the game proceeds as described in Section 2.0.1.

Chapter 3

Computation

3.1 Game-theoretic goal recognition model

With the game defined, the next step is computing a solution to the game. Before defining rational behavior, we first need to discuss the set of strategies. In a sequential game, a pure strategy of a player is a deterministic mapping from the current state and the player's observations/histories leading to the state, to an available action. For the adversary, such observations/histories include its own sequence of prior actions and its target θ. The observer's observations/histories include the adversary's sequence of actions and the observer's sequence of actions. A mixed strategy is a randomized strategy, specified by a probability distribution over the set of pure strategies. The strategies are defined more formally in Section 1.1 and Definition 1.1.1.

We are interested in computing the max-min solution, which is equivalent to the max-min value, min-max value, and Nash equilibrium value of the game. For simultaneous-move games this can usually be solved by formulating a linear program. However, for our sequential game, each pure strategy need to prescribe an action for each possible sequence of observations leading to that state and, as a result, the sets of pure strategies are exponential for both players.

To overcome this computational challenge, we focus on *stationary strategies*, which are strategies that depend only on the current state (for the adversary, also on θ) and not on the history of observations (see Definition 1.1.2). While for stochastic games with complete information, it is known that there always exist an optimal solution that consists of stationary strategies [2], it is an open question whether the same property holds for our setting, which is an incomplete-information game. Nevertheless, there are some heuristic reasons that stationary strategies are at least good approximately optimal solutions: The state (i.e., adversary's location) already capture a large amount of information about the strategic intention of the adversary.

Restricting to stationary strategies, randomized strategies now correspond to a mapping from state to a distribution over actions. We have thus reduced the dimension of the solution space from exponential to polynomial in the size of the graph. Furthermore, our game exhibits the single-controller property: The state transitions are controlled by the adversary only. For complete information stochastic games with a single controller, a *linear programming* (LP) formulation is known [19]. We adapt this LP formulation to our incomplete information setting.

We define $V(\theta, s)$ to be a variable that represents the expected payoff to the observer at state s and with adversary's target begin θ. We use $P(\theta)$ to denote the prior probability of $\theta \in B$ being the adversary's target such that $\sum_{\theta \in B} P(\theta) = 1$. The observer's objective is to find a (possibly randomized) strategy that maximizes his expected payoff given the prior distribution over the target set B, the moves of the adversary, and the adversary's starting location. The following linear program computes the utility of the observer in an max-min solution assuming both players are playing a stationary strategy.

$$\max_{V,\{f_i(s)\}_{i,s}} \sum_{\theta} P(\theta)V(\theta, s_o) \tag{3.1}$$

$$V(\theta, s) \leq \sum_{i\in B} r(s, i, j, \theta) f_i(s) + V(\theta, j) \qquad \forall \theta \in B, \forall s \mid s \neq \theta, \forall j \in \nu(s) \tag{3.2}$$

$$V(\theta, s) = 0 \qquad \text{when } \; s = \theta \tag{3.3}$$

$$\sum_i f_i(s) = 1 \qquad \forall s \tag{3.4}$$

$$f_i(s) \geq 0 \qquad \forall s, i \tag{3.5}$$

In the above linear program, (4.1) is the objective of the observer. The $f_i(s)$'s represent the probability of the observer taking an action $i \in B$ given the state s. To ensure that the probability distribution is well defined at each state of the games, (4.6) and (4.8) impose the standard sum-equal-to-one and non-negative conditions on the probability of playing each action $i \in B$. The Bellman-like inequality (4.3) bounds the expected value for any state using expected values of next states plus the expected current reward, assuming the adversary will choose the state transition that minimizes the observer's expected utility. Finally, (4.4) specifies the base condition when the adversary has reached their destination and the game ends. The size of the linear program is polynomial in the size of the graph.

The solution of this linear program prescribes a randomized stationary strategy $f_i(s)$ for the observer and, from the dual solutions, one can compute a stationary strategy for the adversary. In more detail, the dual linear program is

$$\min \sum_{s} t_s \tag{3.6}$$

$$t_s \geq \sum_{\theta,j} \lambda^{\theta}_{s,j} r(s,i,j,\theta) \qquad \forall s, i \tag{3.7}$$

$$I_{s=s_0} P(\theta) + \sum_{s' \neq \theta : s \in \nu(s')} \lambda^{\theta}_{s',s} = \sum_{j \in \nu(s)} \lambda^{\theta}_{s,j} \qquad \forall \theta \in B, \forall s \neq \theta \tag{3.8}$$

$$\lambda^{\theta}_{s,j} \geq 0 \qquad \forall \theta, s, j \tag{3.9}$$

where $I_{s=s_0}$ is the indicator that equals 1 when $s = s_0$ and 0 otherwise. The dual variables $\lambda^{\theta}_{s,j}$ can be interpreted as the probability that adversary type θ takes the edge from s to j. These probabilities satisfies the flow conservation constraints (3.8): given θ, the total flow into s (the left hand side) is equal to the probability that type θ visits s, which should equal the total flow out of s (the right hand side). The variables t_s can be interpreted as the contribution to defender's utility from state s, assuming that the defender is choosing an optimal action at each state (ensured by constraint (3.7)).

Given the dual solutions $\lambda^{\theta}_{s,j}$, we can compute a stationary strategy for the adversary: let $\pi(j|\theta, s)$ be the probability that the adversary type θ chooses j at state s. Then for all $\theta \in B$ and $s \neq \theta$, $\pi(j|\theta, s) = \frac{\lambda^{\theta}_{s,j}}{\sum_{j' \in \nu(s)} \lambda^{\theta}_{s,j'}}$. It is straightforward to verify that by playing the stationary strategy π, the adversary type θ will visit each edge (s, j) with probability $\lambda^{\theta}_{s,j}$.

Lemma 3.1.1. *Given a stationary strategy for the defender, there exists a best response strategy for the adversary that is also a stationary strategy.*

Sketch. Given a stationary defender strategy $f_i(s)$, each adversary type θ now faces a Markov Decision Process (MDP) problem, which admits a stationary strategy as its optimal solution. □

More specifically, since the state transitions are deterministic and fully controlled by the adversary, each type θ faces a problem of determining the shortest path from s_0 to θ, with the cost of each edge (s, j) as $\sum_{i \in B} f_i(s) r(s, i, j, \theta)$. Looking into the components of $r(s, i, j, \theta)$, since the adversary reward u^θ for reaching target θ occurs exactly once at the target θ, it can be canceled out and the problem is equivalent to the shortest path problem from s_0 to θ with edge cost $d + f_\theta(s) q$. Since edge costs are nonnegative the shortest paths will not involve cycles.

What this lemma implies is that if the defender plays the stationary strategy prescribed by the LP (4.1), the adversary cannot do better than the value of the LP by deviating to a non-stationary strategy.

Corollary 3.1.1. *If the defender plays the stationary strategy $f_i(s)$ given by the solutions of LP* (4.1)*, the adversary's stationary strategy π as prescribed by LP* (3.6) *is a best response, i.e., no non-stationary strategies can achieve a better outcome for the adversary.*

While it is still an open question whether the defender has an optimal strategy that is stationary, we have shown that if we restrict to stationary strategies for the defender, it is in the best interest of the adversary to also stick to stationary strategies and our LP (4.1) do not overestimate the value of the game.

3.1.1 Belief Update

Using the dual LP, it is possible for the observer to abandon their strategy, and instead update their belief every time-step. Such a strategy would no longer be stationary, but if the observer could improve their performance then it would call the equilibrium of the stationary strategies into question. To do this, the observer begins with a probability distribution Υ describing their belief as to which target the adversary intends to reach. Each potential target has a matching $\upsilon \in \Upsilon$ which represents the target's likelihood of being the adversary's true target. This probability distribution begins equal to P, the distribution provided by nature at the start of the game. Each time the adversary makes a move, the observer refers to the dual solution, and multiplies the each value in their distribution by the corresponding $\lambda^{\theta}_{s,j}$ from the dual LP. Then, the observer makes a guess based on the largest value in Υ. We will conduct an experiment later on to test the performance of this non-stationary strategy.

3.1.2 Game-theoretic goal recognition design model

One can solve this GTGRD problem by brute-force, i.e., try every subset of edges to block and then for each case solve the resulting LP. The time complexity of this approach grows exponentially in K. Instead, we can encode the choice of edge removal as integer variables added to the LP formulation, resulting in a mixed-integer program (MIP). For example, we could replace (4.3) with

$$V(\theta, s) \leq \sum_{i \in B} r(s, i, j, \theta) f_i(s) + V(\theta, j) + Mz(s, j) \tag{3.10}$$

where M is a positive number, and $z(s, j)$ is a 0-1 integer variable indicating whether the action/edge from s to j is blocked. M thus represents the penalty that the attacker incurs if he nevertheless chooses to take the edge from s to j while it is blocked. By making M sufficiently large, we can make the actions of crossing a blocked edge dominated and therefore effectively removing the edges that we block. We also add the constraint $\sum_{s,j} z(s, j) \leq K$.

Dual-based greedy heuristic.

The MIP approach scales exponentially in the worst case as the size of the graph and K grows. We propose a heuristic method for selecting edges to block. We first solve the LP for goal recognition and its dual. In particular, we look at the dual variable $\lambda^{\theta}_{s,j}$ for the constraint (4.3). This dual has the standard interpretation as the *shadow price*: it is the rate of change to the objective if we infinitesimally relax constraint (4.3).

Looking at the MIP, in particular constraint (3.10), we see that by blocking off an action from s to j we are effectively relaxing the corresponding LP constraints (4.3) indexed by θ, s, j for all $\theta \in B$. These are the adversary's incentive constraints for going from s to j, for all adversary types θ.

Utilizing the shadow price interpretation of the duals, the sum of the duals corresponding to the edge from s to j: $\sum_{\theta \in B} \lambda^{\theta}_{s,j}$ gives the rate of change to the objective (i.e. defender's expected utility) if the edge (s, j) is blocked by an infinitesimal amount. Choosing the edge that maximizes this, $\arg\max_{s,j} \sum_{\theta \in B} \lambda^{\theta}_{s,j}$ we get the maximum rate of increase of our utility. These rates of changes hold only when the amount of relaxation (i.e., M) is infinitesimal. However, in practice we can still use this as a heuristic for choosing edges to block.[1]

When $K > 1$, we could choose the K edges with the highest dual sums. Alternatively, we can use a greedy approach: pick one edge with the maximum dual sum, place a block on the edge and solve the updated LP for goal recognition, and pick the next edge using the updated duals, and repeat. In our experiments, the latter greedy approach consistently achieved significantly higher expected utilities than the former. Intuitively, by re-solving the

[1] Another perspective: from the previous section we see that $\lambda^{\theta}_{s,j}$ is the probability that adversary type θ traverses the edge s, j. Then if the adversary and defender do not change their strategies after the edge (s, j) is blocked, the defender would receive an additional utility of $M \sum_{\theta \in B} \lambda^{\theta}_{s,j}$ from the adversary's penalty for crossing that edge.

LP after adding each edge, we get a more accurate picture of the adversary's adaptations to the blocked edges. Whereas the rates of changes used by the former approach are only accurate when the adversary do not adapt at all to the blocked edges (see footnote 1). Our greedy heuristic is summarized as follows.

- for $i = 1 \dots K$:
    - Solve LP (4.1), updated with the current blocked edges. If edge (s, j) blocked, the corresponding constraint (4.3) indexed s, j, θ for all θ are modified so that M is added to the right hand side. Get the primal and dual solutions.
    - Take an edge $(s^*, j^*) \in \arg\max_{s,j} \sum_{\theta \in B} \lambda^{\theta}_{s,j}$, and add it to the set of blocked edges.
- return the set of blocked edges, and the primal solution of the final LP as the defender's stationary strategy.

3.2 Experiments

Experiments were run on a machine using OSX Yosemite version 10.10.5, with 16 GB of ram and a 2.3 GHz Intel Core i7 processor, and were conducted on grid environments such as the one seen in Figure 3.1. In these environments, the adversary is allowed to move to adjacent nodes connected by an edge. S denotes the starting location of the adversary while $T1$ and $T2$ denote the locations of two potential targets.

In Figure 3.1, targets $T1$ and $T2$ each have a equal likelihood of being the adversary's intended target. The adversary's timestep penalty d and completion reward u^{θ} are both set to 0. The defender's reward for correctly guessing the adversary's intended target q is set

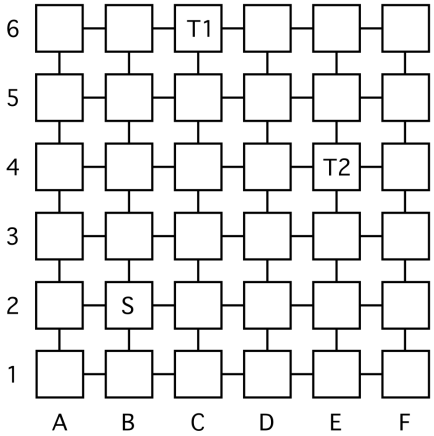

Figure 3.1: An Instance of GTGR/GTGRD Games Used in Experiments.

to 10. The attacker penalty value for crossing an edge penalized by the observer is set to 10. The observer is permitted to penalize 3 edges.

3.2.1 A Comparison of MIP and Greedy Solutions

As seen in Figure 3.2 and Figure 3.3, the mixed integer program and greedy heuristic can yield different results. The mixed integer program yields an expected outcome of 43.3 for the observer, while utilizing the greedy heuristic yields an outcome of 40.0 for the observer. The default expected outcome for the observer (in which no edges are penalized) is 30.0. The following experiments averaged the results of similar grid problems.

3.2.2 Running Time and Solution Quality

Results from the following experiments were averaged over 1000 grid environments. For each experiment, the adversary's timestep penalty d and completion reward u^{θ} were set to 0. For each environment, the starting location of the adversary and all targets are placed

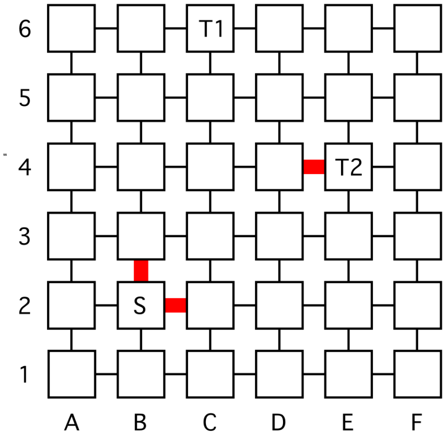

Figure 3.2: MIP Solution

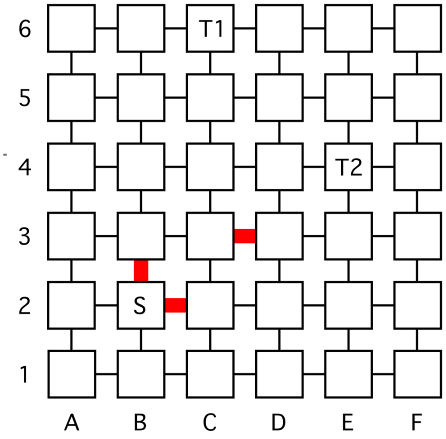

Figure 3.3: Greedy Solution

randomly on separate nodes. Additionally, each target θ is assigned a random probability $P(\theta)$ such that $\sum_{\theta \in B} P(\theta) = 1$. In all of our figures below, the greedy heuristic for the GTGRD is graphed in green, the MIP is graphed in blue, and the default method (LP) for GTGR is graphed in red, in which the game is solved with no penalized edges. The defenders reward for correctly guessing the adversary's intended target q was set to 10. The attacker penalty value for crossing an edge penalized by the observer was set to 10. Each game, the observer was permitted to penalize 2 edges.

Various Potential Target Sizes

In this set of experiments, we want to investigate the effect of different potential target sizes (i.e., $|B|$) to the running time (Figure 3.4) and solution quality (Figure 3.5) of our algorithms. The results are averaged over 1000 simulations of 6 by 6 grids. Each game, the observer was permitted to penalize 2 edges.

As indicated in Figure 3.4, the MIP running time increases exponentially while the

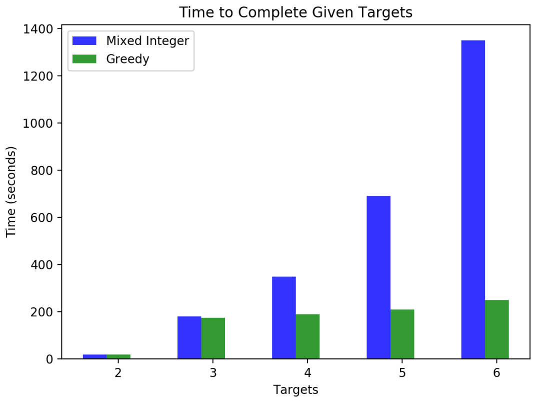

Figure 3.4: Average time given targets.

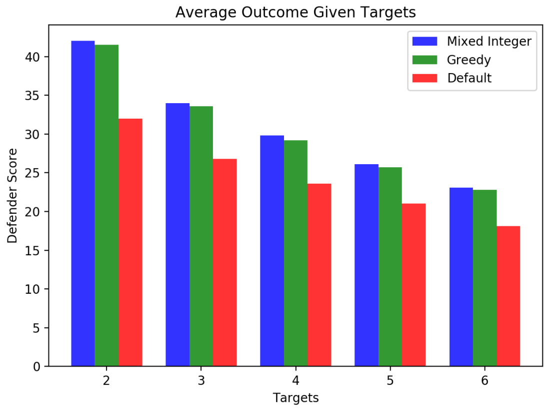

Figure 3.5: Average outcome given targets.

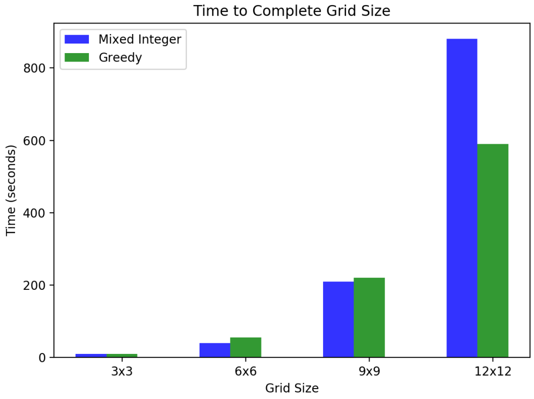

Figure 3.6: Average time given size.

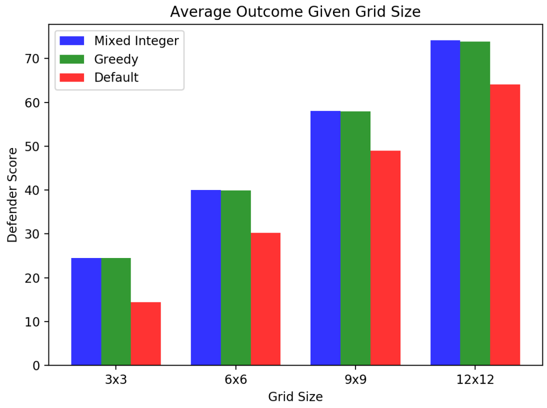

Figure 3.7: Average outcome given size.

greedy heuristic running time remains sublinear as we increase the number of potential targets. Moreover, the solution quality (measured by defender's utility) as seen in Figure 3.5 suggests that MIP's solution is closely aligned with our greedy heuristics. This gives evidence that our greedy heuristic provides good solution quality while achieving high efficiency. Note that It is no surprise that the defender's utility is higher in the GTGRD setting compared to those of GTGR.

Various Instance Sizes

In this set of experiments, we investigate the effect of different instance sizes (i.e., grids) to the running time (Figure 3.6) and solution quality (Figure 3.7) of our algorithms.

Unlike our earlier observations on various target sizes, the average running times for both the MIP and our greedy heuristic increase significantly as we increase the instance sizes (see Figure 3.6). This is not surprising as now we have more variables and constraints in the integer programs. Despite this, the defender's utilities generated by greedy heuristic are relatively similar to to those generated using MIP (see Figure 3.7).

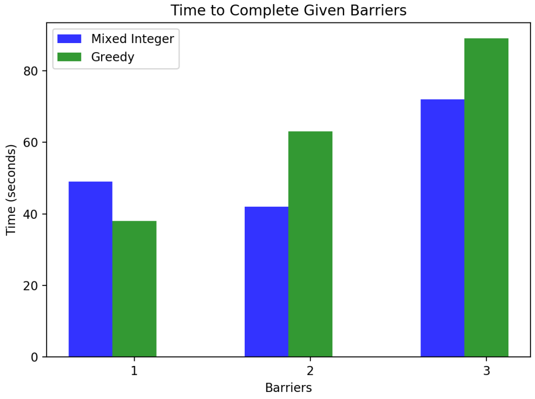

Figure 3.8: Average time given penalized edges.

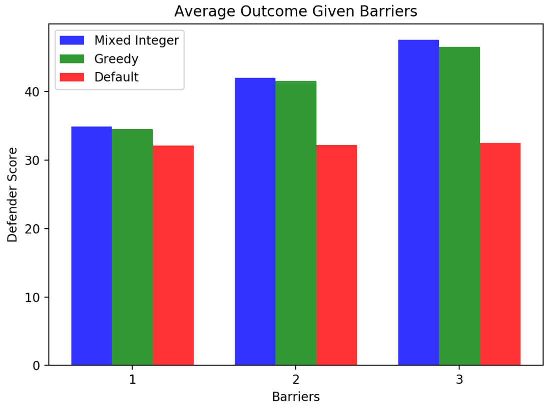

Figure 3.9: Average outcome given penalized edges.

Various Number of Barriers/Blocks

In this set of experiments, we want to investigate the effect of different number of barriers (i.e., K) to the running time (Figure 3.4) and solution quality (Figure 3.5) of our algorithms

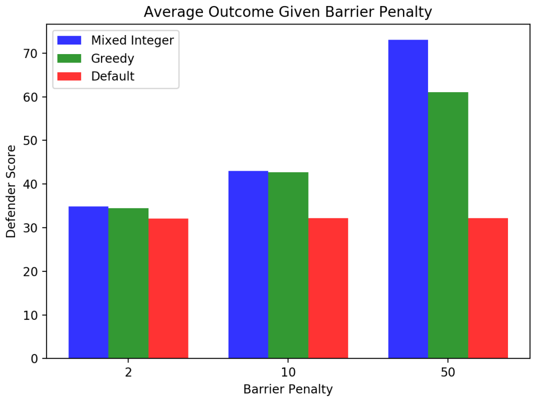

Figure 3.10: Average outcome given penalty value.

in the GTGRD models. The results are averaged over 1000 simulations of 6 by 6 grids.

It turns out that as we increase the number of barriers, the running times of our greedy heuristic are longer than the MIP as shown in Figure 3.8. Nonetheless, as in the earlier experiments, both algorithms have similar solution quality.

Various Edge Penalties

Next, consider the effect of different edge penalties to the solution quality of our greedy heuristic. The results are averaged over 1000 simulations of 6 by 6 grids. As indicated in Figure 3.10, the solution gap between the MIP and greedy heuristic as we increase the edge penalty.

Belief Update Comparison

Finally, we will measure the performance of a non-stationary strategy using belief update, against the original LP. For this experiment, results were averaged over 1000 grids, each with a width and height of 9 nodes. The adversary's starting location, and 2 targets were placed on random distinct nodes for each grid. The defender's guess reward was set to 10, with all other rewards and penalties set to 0. No barriers were placed for these tests.

Both the stationary strategy and the non-stationary strategy from belief update produced an average observer score of precisely 47.543. Surprisingly, using belief update did not increase or decrease the observer's average score at all. This does not prove that the stationary strategy is optimal for the observer, only that a non-stationary strategy from belief update seems incapable of doing any better in this scenario.

Chapter 4

Partially Observable Environments

4.1 Introduction

Until now, both the GTGR and GTGRD models have given the observer full knowledge of the adversary's state for the entirety of the game. In real-world environments, observers may not have perfect information regarding the states and actions of an adversary.

To accommodate for scenarios with incomplete information for the adversary, we introduce a partially observable variant of the GTGR scenario. In partially observable scenarios, the rules of the game remain largely unchanged, except for addition of shadow states." The observer can not discern the current state of the adversary, while the adversary occupies a shadow state. When the adversary enters an observable portion of the graph, the observer will become aware of the adversary's position once more.

Figure 4.1 illustrates a partially observable environment. Visible states, in which the observer can see the adversary white. Shadow states, in which the adversary is hidden from the observer, are black. The agent starts the game in state S. When the adversary moves to states 4 ,5, 6, or 7, the observer is unable to determine their position until the adversary

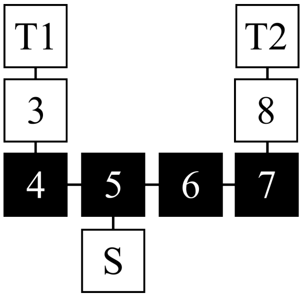

Figure 4.1: A partially observable graph.

re-enters a visible portion of the graph. We will examine two solutions to the partially observable model, both of which involving linear programming.

4.2 The Whale Method

The first method of solving partially observable environments, which we will call the "Whale Method," will utilize disjoint sets of shadow states. We call these sets of shadow states "shadow sets." We say that two shadow states belong to the same shadow set, if the adversary can travel between the two without entering an observable state.

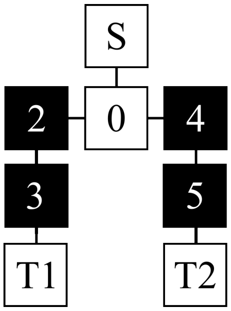

Figure 4.2: A partially observable graph with two shadow sets.

The graph in Figure 4.2 has two disjoint shadow sets, one composed of states 2 and 3, the other composed of states 4 and 5. In using the Whale Method, the observer treats each shadow set a single state, which we will call a "whale state." We can identify the single shadow set in Figure 4.1, composed of states 4, 5, 6, and 7. In using the Whale method, the observer treats each state in the shadow set as the same state. While the adversary may require several turns to travel among states 4, 5, 6, and 7, the adversary will act as if the has decided to remain stationary in the newly created state W seen in Figure 4.3

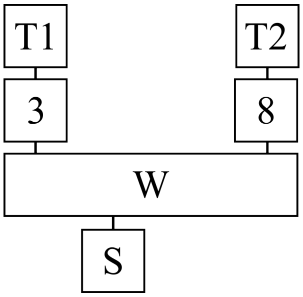

Figure 4.3: A partially observable graph with a single state in place of shadow states.

We can add the following to the mixed integer program to accommodate for partially observable environments when using the Whale method.

$$\max_{V,\{f_i(s)\}_{i,s}} \sum_{\theta} P(\theta)V(\theta, s_o) \tag{4.1}$$

$$V(\theta, s) \leq \sum_{i\in B} r(s,i,j,\theta)f_i(s) + V(\theta, j) \qquad \forall\theta \in B, \forall s \mid s \neq \theta, s \notin \Omega, \forall j \in \nu(s) \tag{4.2}$$

$$V(\theta, s) \leq \sum_{i\in B} r(s,i,j,\theta)f_i(w) + V(\theta, j) \qquad \forall\theta \in B, \forall s \mid s \neq \theta, s \in \Omega, \forall j \in \nu(s) \tag{4.3}$$

$$V(\theta, s) = 0 \qquad \text{when } s = \theta \tag{4.4}$$

$$\sum_{i} f_i(s) = 1 \qquad \forall s \tag{4.5}$$

$$\sum_{i} f_i(w) = 1 \qquad \forall w \tag{4.6}$$

$$f_i(s) \geq 0 \qquad \forall s, i \tag{4.7}$$

$$f_i(w) \geq 0 \qquad \forall w, i \tag{4.8}$$

We let Ω denote the set of all shadow states, and w denote the whale state the observer knows the adversary to be occupying. An observer action for any whale state w is written as $f_w(s)$. These changes to the linear program require the observer to take the same action for each turn the adversary spends in a particular shadow set. The performance of the Whale method will be examined in a later section.

4.3 The Transmogrification Method

When using the Whale method, the observer ignores some of the information available to them. The Whale method does not account for where the adversary entered a shadow set, or how long the adversary has remained hidden. The "Transmogrification Method" takes this information into account, by generating a fully observable environment from a partially observable environment. To do this, the observer must make some basic assumptions about the adversary's strategy. The following lemmas and corollary assume the agent is playing optimally against the observer's stationary strategy.

Lemma 2. *If the adversary's target does not lie within a shadow set, the adversary will eventually exit the shadow set.*

Proof (Sketch). As mentioned previously, the game could theoretically go on forever. But because of the potential per-timestep cost of d and the observer's predictions, any sufficiently long path for the adversary would be dominated by the strategy of taking the shortest path to their target θ. If the adversary never leaves the set of shadow states, then the game will go on forever. Thus, the adversary will eventually leave a shadow set.

Corollary 2. *An optimal agent occupying a state in a shadow set will take a shortest path to the exit state of their choosing.*

Proof (Sketch). We established that an optimal adversary in a shadow set must exit that shadow set. Each unnecessary turn the adversary spends in a shadow state invites the observer to guess their intended target. Thus, it is in the observer's interest to reach their chosen exit as quickly as possible.

With Corollary 2, the observer can generate a new graph to play on.

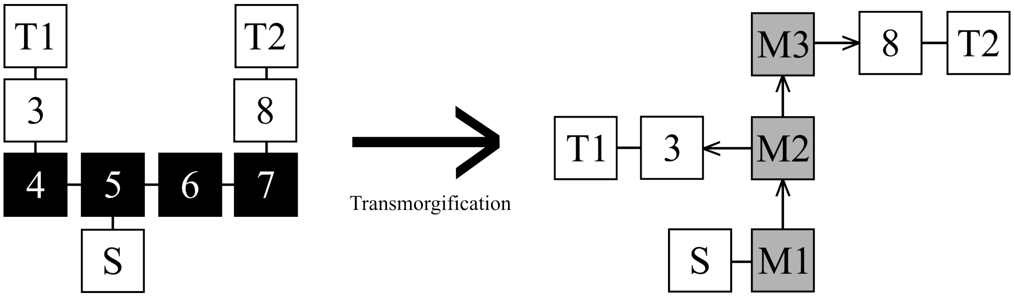

Figure 4.4: The partially observable environment from Figure 4.1 (left) transmogrified into an, fully observable graph.

Figure 4.4 illustrates a transmogrified version of the graph from Figure 4.1. The grey nodes in the transmogrified graph are introduced to represent the number of turns the adversary has remained hidden from the observer. For instance, if the adversary has been hidden for two turns, then the observer would see the adversary as occupying state $M2$ in the transmogrified graph. Take note that the transmogrified graph now includes directed edges, because we are (hopefully) not dealing with a time traveling agent. To generate these graphs, we examine each shadow set. For each shadow set, we examine each "entrance state" connected by an edge to the shadow state. We then compute the shortest path between every two entrance states for that shadow state. By Corollary 2, we assume the length of the longest shortest path will be the maximum number of turns an optimal agent will spend within the shadow set. We then create create the same number of memory nodes with directed edges from one to the next (see states $M1, M2, M3$ from state S in Figure 4.4). Then we create an edge from each entrance state, to the memory state corresponding to the length of the shortest path between the entrance state, and the original entrance state from which we spawned the memory states. For example, because a adversary would have to two shadow states on their journey from state S to state 3 in Figure 4.1, we connect state $M2$ to state 3 in Figure 4.4. Note, that the post transmogrification graph in Figure 4.4 is actually incomplete, because memory states have only been spawned from the entrance state S and not from entrance states 3 or 8. Because an optimal agent has no reason to backtrack in this particular scenario, the memory nodes for 3 and 8 were omitted to keep the graph simple.

Next, let us examine how a game might play out from the adversary's perspective. Using the environment from Figure 4.1, let us assume the adversary was assigned target $T1$. Because the graph is rather limiting, the adversary takes the shortest path from starting state S to the target $T1$. Figure 4.5 illustrates the adversary's journey, each turn market

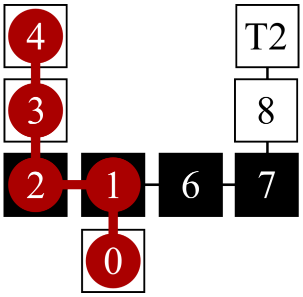

Figure 4.5: The adversary's path to target $T1$.

with a red circle. Note that on turns 1 and 2, the observer would lose sight of the attacker until turn 3. Now, let us examine what the same scenario would look like from the observer's prospective, when using the transmogrification method.

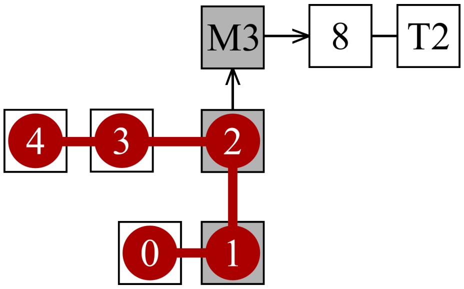

Figure 4.6: The adversary's path to $T1$ from the observer's perspective.

Because the adversary takes two turns within the shadow set, the observer sees the adversary as moving from state $M1$ to state $M2$, before exiting the shadow set on turn 3.

4.4 Experiments

As with previous experiments, all tests were run on a machine using OSX Yosemite version 10.10.5, with 16 GB of ram and a 2.3 GHz Intel Core i7 processor. First, we will use the simple example from Figure 4.1 to compare the performance of the Whale, and Transmogrification methods. Additionally, we will compare the performance against a best case solution, in which we turn all shadow states into standard states. The best case solutions measures how the observer would perform if their strategy effectively negated the shadow states. Graphs were displayed using the Python NetworkX library. Arrowheads on directed edges were added manually for clarity. The starting state is displayed in green, target states are displayed in red, shadow states are displayed in purple, and default states are displayed in cyan.

After transmogrifying the graph, the scenario becomes more complex, but allows the observer to play a fully observable game. There graph will no longer have any shadow states (purple). All states labeled with values greater than 1000 are nodes that have been added to represent turns hidden within the shadow set. Note that the graph in Figure 4.8 will have more nodes than the previously seen illustration of the transmogrified graph in Figure 4.4, because we spawn memory states for every entry state, and not just the starting state.

For this game, the adversary is not penalized for timesteps taken. The reward for reaching the target is 0. The reward the observer receives for each correct guess was set to 1. The point value for each method at the end of the game equates to the number of correct guesses the observer can expect to make. At the start of the game, the adversary has a 75% chance of being assigned target 1, and a 25% of being assigned target 2.

No Shadow States	3.75 Correct Guesses
Whale	3.25 Correct Guesses
Transmogrification	3.50 Correct Guesses

Table 4.1: Performance results from simple environment.

If a strategy were to effectively remove the shadow states from the board, the observer

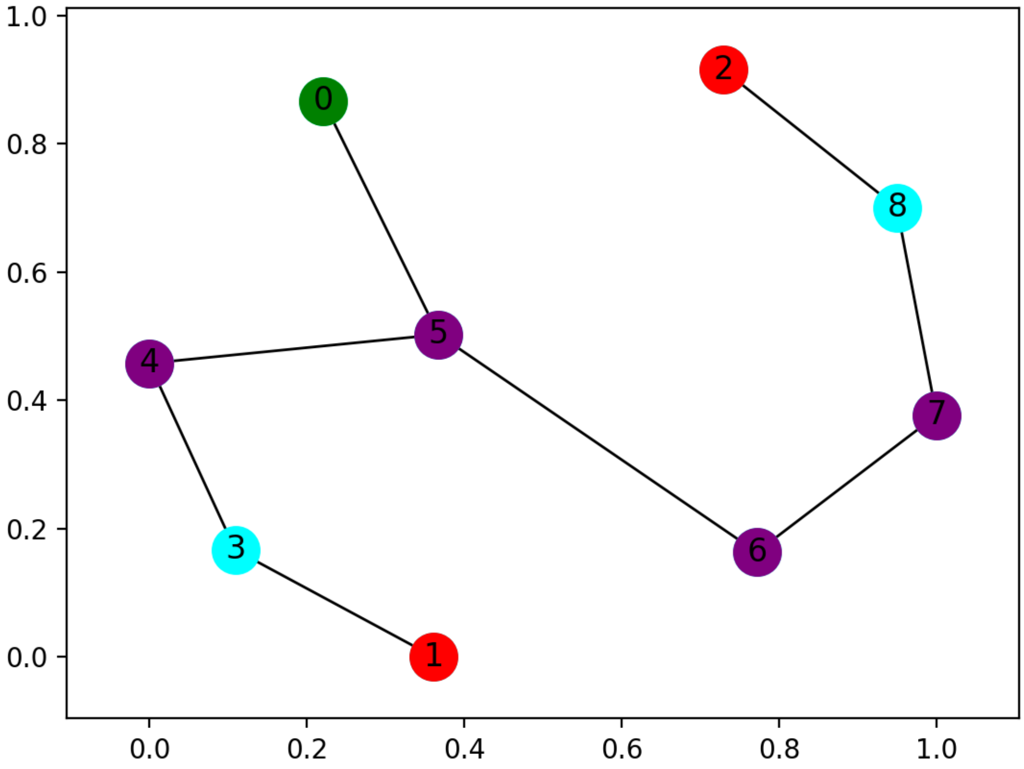

Figure 4.7: A simple scenario using NetworkX.

could expect to make 3.75 correct guesses over the course of the game. Thus, we could not expect the whale or transmogrification methods to perform any better. Using the whale methods, in which all shadow states a mushed together into one whale of a state, the observer can expect to make 3.25 correct guesses. While the transmogrification method does not allow the observer to effectively see through shadow states, the method still outperforms the whale method with a score of 3.50. Next, we will test these methods against each other

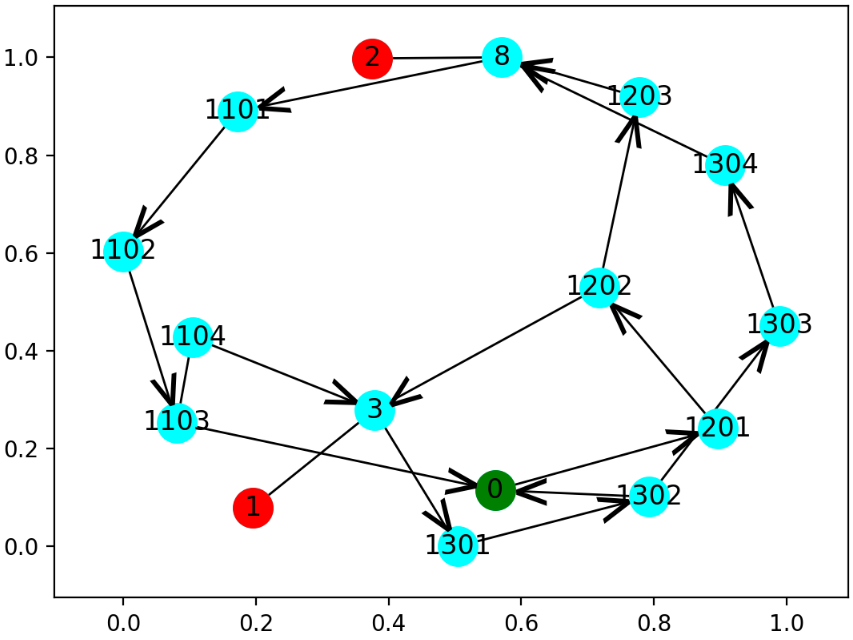

Figure 4.8: The transmogrified graph.

in a more complex scenario.

With the simple example out of the way, we move to a more complex environment which offers the adversary more freedom in approaching their target. For the next set of tests, we place the adversary starting state, and 3 potential targets on non-shadow states (marked in white) in Figure 4.9. We then create a random probability distribution, and solve the game with both methods. After ten-thousand iterations, the scores are averaged and compared.

Though the transmogrification method does not yield what the observer could score without shadow states, it still outperforms the whale method.

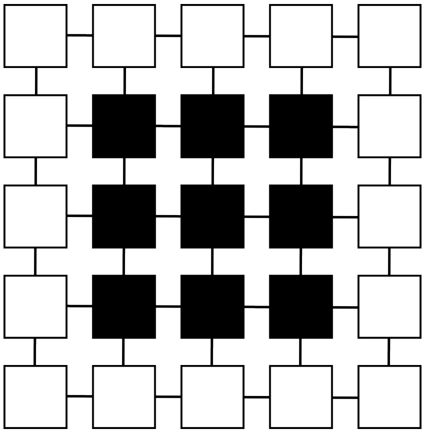

Figure 4.9: Complex testing environment.

No Shadow States	2.65 Correct Guesses
Whale	2.41 Correct Guesses
Transmogrification	2.59 Correct Guesses

Table 4.2: Average method performance in complex environment.

Chapter 5

Conclusion

Motivated by the goal recognition (GR) and goal recognition design (GRD) problems in the artificial intelligence (AI) planning domain, we introduced and studied two natural variants of the GR and GRD problems with strategic agents. We considered game-theoretic (GT) scenarios where a malicious adversary is aiming to cause damage to some target in an (physical or virtual) environment monitored by a defender. We modeled GTGR and GTGRD settings as zero-sum stochastic games with incomplete information regarding the adversary's intended target. We presented two solutions to the GTGRD setting. The mixed integer program gave optimal results when restricting the observer to stationary strategies. The dual-based greedy heuristic was capable of achieving similar performance with far better scalability. We also introduced a partially observable variant of the GTGR setting with imperfect information regarding the agent's location. The Whale method for approaching partially observable scenarios offered an inexpensive solution. The Transmogrification, though incredibly expensive offered better results. Though currently impractical, the Transmogrification method could become viable with the addition of selective pruning.

Bibliography

[1] S. Braynov. Adversarial planning and plan recognition: Two sides of the same coin. In *Proceedings of the Secure Knowledge Management Workshop*, 2006.

[2] Drew Fudenberg and Jean Tirole. Game theory, 1991.

[3] Christopher Geib and Mark Steedman. On natural language processing and plan recognition. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1612–1617, 2007.

[4] N. Le Guillarme, A.-I. Mouaddib, X. Lerouvreur, and S. Gatepaille. A generative game-theoretic framework for adversarial plan recognition. In *Proceedings of the Workshop on Distributed and Multi-Agent Planning*, 2015.

[5] Peter Jarvis, Teresa Lunt, and Karen Myers. Identifying terrorist activity with AI plan recognition technology. *AI Magazine*, 26(3):73–81, 2005.

[6] W. Lewis Johnson. Serious use of a serious game for language learning. *International Journal of Artificial Intelligence in Education*, 20(2):175–195, 2010.

[7] Henry A Kautz. *A Formal Theory of Plan Recognition.* PhD thesis, Bell Laboratories, 1987.

[8] Richard Kelley, Liesl Wigand, Brian Hamilton, Katie Browne, Monica Nicolescu, and Mircea Nicolescu. Deep networks for predicting human intent with respect to objects. In *Proceedings of the International Conference on Human-Robot Interaction (HRI)*, pages 171–172, 2012.

[9] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 154–162, 2014.

[10] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design for non-optimal agents. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3298–3304, 2015.

[11] Sarah Keren, Avigdor Gal, and Erez Karpas. Goal recognition design with non-observable actions. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 3152–3158, 2016.

[12] Seung Lee, Bradford Mott, and James Lester. Real-time narrative-centered tutorial planning for story-based learning. In *Proceedings of the International Conference on Intelligent Tutoring Systems (ITS)*, pages 476–481, 2012.

[13] Viliam Lisý, Radek Píbil, Jan Stiborek, Branislav Bosanský, and Michal Pechoucek. Game-theoretic approach to adversarial plan recognition. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 546–551, 2012.

[14] Scott McQuiggan, Jonathan Rowe, Sunyoung Lee, and James Lester. Story-based learning: The impact of narrative on learning experiences and outcomes. In *Proceedings of the International Conference on Intelligent Tutoring Systems (ITS)*, pages 530–539, 2008.

[15] Wookhee Min, Eunyoung Ha, Jonathan Rowe, Bradford Mott, and James Lester. Deep learning-based goal recognition in open-ended digital games. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AI-IDE)*, 2014.

[16] Jean Oh, Felipe Meneguzzi, Katia Sycara, and Timothy Norman. ANTIPA: An agent architecture for intelligent information assistance. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 1055–1056, 2010.

[17] Jean Oh, Felipe Meneguzzi, Katia Sycara, and Timothy Norman. An agent architecture for prognostic reasoning assistance. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2513–2518, 2011.

[18] Jean Oh, Felipe Meneguzzi, Katia Sycara, and Timothy Norman. Probabilistic plan recognition for intelligent information agents: Towards proactive software assistant agents. In *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART)*, pages 281–287, 2011.

[19] T. E. S. Raghavan. Finite-step algorithms for single-controller and perfect information stochastic games. In Abraham Neyman and Sylvain Sorin, editors, *Stochastic Games and Applications*, pages 227–251. Springer Netherlands, 2003.

[20] Miquel Ramírez and Hector Geffner. Plan recognition as planning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1778–1783, 2009.

[21] Miquel Ramírez and Hector Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 1121–1126, 2010.

[22] Miquel Ramírez and Hector Geffner. Goal recognition over POMDPs: Inferring the intention of a POMDP agent. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2009–2014, 2011.

[23] Charles Schmidt, N. Sridharan, and John Goodson. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence*, 11(1–2):45–83, 1978.

[24] Tran Cao Son, Orkunt Sabuncu, Christian Schulz-Hanke, Torsten Schaub, and William Yeoh. Solving goal recognition design using asp. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

[25] Gita Sukthankar, Christopher Geib, Hung Hai Bui, David Pynadath, and Robert P Goldman. *Plan, activity, and intent recognition: Theory and practice.* Newnes, 2014.

[26] Alireza Tavakkoli, Richard Kelley, Christopher King, Mircea Nicolescu, Monica Nicolescu, and George Bebis. A vision-based architecture for intent recognition. In *Proceedings of the International Symposium on Advances in Visual Computing*, pages 173–182, 2007.

[27] Marc Vilain. Getting serious about parsing plans: A grammatical analysis of plan recognition. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 190–197, 1990.

[28] Christabel Wayllace, Ping Hou, William Yeoh, and Tran Cao Son. Goal recognition design with stochastic agent action outcomes. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.