Counterplanning using Goal Recognition and Landmarks

Alberto Pozanco, Yolanda E-Martín, Susana Fernández, Daniel Borrajo
Departamento de Informática, Universidad Carlos III de Madrid
Avda. de la Universidad, 30. 28911 Leganés, Madrid, Spain
apozanco@pa.uc3m.es, yescuder@inf.uc3m.es, sfarregu@inf.uc3m.es, dborrajo@ia.uc3m.es

Abstract

In non-cooperative multi-agent systems, agents might want to prevent the opponents from achieving their goals. One alternative to solve this task would be using counterplanning to generate a plan that allows an agent to block other's to reach their goals. In this paper, we introduce a fully automated domain-independent approach for counterplanning. It combines; goal recognition to infer an opponent's goal; landmarks' computation to identify subgoals that can be used to block opponents' goals achievement; and classical automated planning to generate plans that prevent the opponent's goals achievement. Experimental results in several domains show the benefits of our novel approach.

1 Introduction

In non-cooperative multi-agent systems, agents might want to prevent the opponents from achieving their goals. This task has been named counterplanning [Carbonell, 1981]. Examples of non-cooperative multi-agent domains where this approach can provide great benefits are police controls, cyber security, or real-time strategy games, where this ability has been identified as one of the major challenges for Artificial Intelligence [Ontañón *et al.*, 2013]. Most previous counterplanning approaches are based on domain-dependent solutions, such as rule-based systems [Carbonell, 1978; Rowe, 2003], or Hierarchical Task Networks (HTN) [Willmott *et al.*, 2001].

Recently, there has been increasing interest in the study and generation of agents capable of reasoning about their own and opponents' goals as well as their environment [Cox, 2007]. Some works follow the Goal-Driven Autonomy (GDA) process, which integrates a diverse set of AI components such as HTN planning or explanation generation [Molineaux *et al.*, 2010; Weber *et al.*, 2010]. Other works combine goal recognition and reasoning on actions, applying those techniques in domains such as identifying terrorist activity [Jarvis *et al.*, 2005], air combat [Borck *et al.*, 2015], real-time strategy games [Kabanza *et al.*, 2010], or cyber security [Boddy *et al.*, 2005; Edelkamp *et al.*, 2009; Sarraute *et al.*, 2012; Obes *et al.*, 2013; Hoffmann, 2015]. Again, these approaches are domain-dependent. On the goal recognition side, they use plan [Kabanza *et al.*, 2010], rules [Carbonell, 1978] or behavior [Borck *et al.*, 2015] libraries to detect their opponent's goals. On the action reasoning side, they use stored policies [Carbonell, 1981], ask for human guidance following a mixed-initiative paradigm [Jarvis *et al.*, 2005], or require heavy knowledge engineering processes such as HTN based approaches [Willmott *et al.*, 2001].

In this paper we present a fully automatic domain-independent approach for counterplanning. This approach is based on: goal recognition, landmarks, and classical automated planning. Goal recognition aims to infer an agent's plan or goals from a set of observations. In general, the observed agent can be cooperative or competitive. We use this technique to infer an opponent's goals. Fact landmarks are propositions that must be true in all valid solution plans [Hoffmann *et al.*, 2004]. We use landmarks to identify subgoals that can be used to block the opponent's goal achievement. Classical automated planning aims to generate a sequence of actions, namely a plan, which achieves some goals from an initial state. We use it to generate plans that prevent the opponent's goal achievement.

The main idea of this novel approach is to: (1) quickly identify the actual opponent's goal g using planning-based goal recognition techniques; (2) compute the set of landmarks involved in the achievement of g; (3) select a *counterplanning landmark*, which is the first landmark where the opponent could be blocked; and (4) generate a plan to achieve the counterplanning landmark, and therefore to block the opponent's goal achievement. This approach shows how an opponent can be effectively blocked in different non-cooperative domains.

The rest of the paper is organized as follows. In the next section we review the basic notions of classical planning, goal recognition, and landmarks. In Section 3 we introduce our fully automatic domain-independent counterplanning approach, which includes the quick detection of goals using goal recognition, and the landmark's computation to identify relevant counterplanning landmarks. In Section 4 we present an empirical study, and in Section 5 we discuss future work.

2 Background

2.1 Automated Planning

Automated Planning is the task of choosing and organizing a sequence of actions such that, when applied in a given initial state, it results in a goal state [Ghallab *et al.*, 2004]. Formally, a single-agent STRIPS planning task can be defined as a tuple $\Pi = \langle F, A, I, G\rangle$, where F is a set of propositions, A is a set of instantiated actions, $I \subseteq F$ is an initial state, and $G \subseteq F$ is a set of goals. Each action $a \in A$ is described by a set of preconditions (pre(a)), which represent literals that must be true in a state to execute an action, and a set of effects (eff(a)), which are the literals that are added (add(a) effects) or removed (del(a) effects) from the state after the action execution. The definition of each action might also include a cost $c(a)$ (the default cost is one). The execution of an action a in a state s is defined by a function γ such that $\gamma(s, a) = (s \setminus \text{del}(a)) \cup \text{add}(a)$ if pre(a)$\subseteq s$, and s otherwise (it cannot be applied). The output of a planning task is a sequence of actions, called a plan, $\pi = (a_1, \ldots, a_n)$. The execution of a plan π in a state s can be defined as:

$$\Gamma(s,\pi) = \begin{cases} \Gamma(\gamma(s,a_1),(a_2,\ldots,a_n)) & \text{if } \pi \neq \emptyset \\ s & \text{if } \pi = \emptyset \end{cases}$$

A plan π is valid if $G \subseteq \Gamma(I, \pi)$. The plan cost is commonly defined as $c(\pi) = \sum_{a_i \in \pi} c(a_i)$. We will use the function PLANNER(Π) to refer to an algorithm that computes a plan π from a planning task Π.

2.2 Goal Recognition

Goal Recognition is the task of inferring another agent' goals through the observation of its interactions with the environment. The problem has captured the attention of several computer science communities [Albrecht *et al.*, 1997; Geib and Goldman, 2009; Sukthankar *et al.*, 2014]. Among them, planning-based goal recognition approaches have been shown to be a valid domain-independent alternative to infer agents' goals [Ramírez and Geffner, 2009; 2010; 2011; Pattison and Long, 2010; E-Martín *et al.*, 2015; Vered and Kaminka, 2017; Pereira *et al.*, 2017]. Ramírez and Geffner [2010] developed an approach that assumes observations are actions, and formally defined a planning-based goal recognition problem as:

Definition 1 (Goal Recognition Problem) *A goal recognition problem is a tuple $T = \langle P, \mathcal{G}, O, Pr\rangle$ where $P = \langle F, A, I\rangle$ is a planning domain and initial conditions, $\mathcal{G}$ is the set of possible goals G, $G \subseteq F$, $O = (o_1, ..., o_m)$ is an observation sequence with each o_i being an action in A, and Pr is a prior probability distribution over the goals in $\mathcal{G}$.*

The solution to a goal recognition problem is a probability distribution over the set of goals $G \in \mathcal{G}$ giving the relative likelihood of each goal. In this work we assume that Pr is uniform. We will use the function RECOGNIZEGOALS($F, A, I, \mathcal{G}, O$) to refer to an algorithm that solves the goal recognition problem. This function returns a list of tuples in the form of $\langle$goal, probability$\rangle$ for each goal in $\mathcal{G}$.

2.3 Landmarks

In Automated Planning, landmarks were initially defined as sets of propositions that have to be true at some time in every solution plan [Hoffmann *et al.*, 2004]. Formally:

Definition 2 (Fact Landmark) *Given a planning task $\Pi = \langle F, A, I, G\rangle$, a formula $L_\Pi \subset F$ is a fact landmark of Π iff L_Π is true in some state along all valid plans executions that achieve G from I.*

This definition was later extended to include action landmarks [Richter and Westphal, 2010]. We will use the function EXTRACTLANDMARKS(F, A, I, G) to refer to an algorithm that computes a set of landmarks $\mathcal{L}_\Pi$ from a planning task Π.

3 Domain-Independent Counterplanning

We first formalize the two actors involved in a counterplanning problem as planning agents.

Definition 3 (Seeking agent) *A seeking agent ϕ is an agent that has an associated planning task $\Pi_\phi = \langle F_\phi, A_\phi, I_\phi, G_\phi\rangle$, and pursues its goal G_ϕ by following a plan π_ϕ computed from Π_ϕ.*

Definition 4 (Preventing agent) *A preventing agent α is an agent that has an associated planning task $\Pi_\alpha = \langle F_\alpha, A_\alpha, I_\alpha, G_\alpha\rangle$.*

G_α is initialized to $\emptyset$. Then, Algorithm 1 (described later) computes a set of goals to be used for the counter-planning task. There can be varied relations between Π_ϕ and Π_α, and the information that one agent has from the other. For instance, the actions that both agents can perform could be the same $A_\phi = A_\alpha$, or totally different $A_\phi \cap A_\alpha = \emptyset$. They could also have different or equal observations of the world. In this work, we make the following assumptions:

- ϕ's model is known by α except for its goal G_ϕ. In most real-world domains that we have selected for potential applications (e.g. police control, cyber security, strategy games, ...), both F_ϕ, I_ϕ, and A_ϕ can be assumed to be known;
- as in most literature on goal reasoning, α knows a set of potential goals, $\mathcal{G}_\phi \subset F_\phi$, ϕ could be trying to achieve;
- deterministic action outcomes and full observability of those actions by α;
- both agents can follow optimal or suboptimal strategies to reach their goals;
- both agents stick to their plans. In other words, they do not replan or change their goals during execution; and
- the temporal duration of an action $a_i \in A$ is determined by its cost $c(a_i)$.[1]

Since both agents operate in a common environment, the execution of their actions affects the shared environment. Therefore, we assume any state of the environment s can be defined in terms of the set of propositions F_e ($s \subseteq F_e$), such that $F_\phi \cup F_\alpha \subseteq F_e$. Additionally, some propositions must be in $F_\phi \cap F_\alpha$, i.e. they will be observable and modifiable by both agents. The individual execution of actions by any of the two agents in F_e will be based on the respective action sets. Hence, the execution of an action a ($a \in A_\phi \cup A_\alpha$) in a state s is defined using the previous $\gamma(s, a)$. Furthermore, the joint execution of one action per agent in the same time step t can be defined as follows.

[1]In this paper, we assume unit action costs.

Definition 5 (Joint execution of two actions) *Given two actions $a_\phi \in A_\phi$ and $a_\alpha \in A_\alpha$ and an environment state $s \subseteq F_e$, the joint execution of both actions at a time step t results in a new state given by*

$$\gamma_{\phi,\alpha}(s, a_\phi, a_\alpha) = \begin{cases} \gamma(\gamma(s, a_\alpha), a_\phi) & \text{if } a_\phi \text{ not mutex with } a_\alpha \\ \gamma(s, a_\alpha) & \text{otherwise} \end{cases}$$

Similarly, the joint execution of two plans $\Gamma_{\phi,\alpha}(s, \pi_\phi, \pi_\alpha)$ can be defined by the iteration of the joint execution of actions of those plans using $\gamma_{\phi,\alpha}(s, a_\phi, a_\alpha)$. For simplicity, in this paper we assume that the preventing agent always executes its action first when two actions are mutex. We define two mutex actions as follows.

Definition 6 (Mutex actions) *Two actions a_x, a_y executed at a time step t are mutex if any literal in eff(a_x) deletes (adds) any literal in pre(a_y) or if any literal in eff(a_x) deletes (adds) a literal that is added (deleted) in eff(a_y).*

Using these definitions, we can now formally describe a counterplanning task.

Definition 7 (Counterplanning task) *A counterplanning task is defined by a tuple $CP = \langle \Pi_\phi, \Pi_\alpha, \mathcal{G}_\phi, O_\phi \rangle$ where Π_ϕ is the planning task of ϕ, Π_α is the planning task for the preventing agent, $\mathcal{G}_\phi$ is the set of sets of goals that ϕ can potentially pursue, and $O_\phi = (o_1, \dots, o_m)$ is a set of observations by α of the execution of a plan $\pi_\phi = (o_1, \dots, o_m, a_{m+1}, \dots, a_k)$ that solves Π_ϕ.*[2]

We assume that ϕ generates a plan π_ϕ to solve its planning task Π_ϕ prior to counterplanning, and that plan (as well as its corresponding goals) is unknown for α. Then, at some time step m of the execution of π_ϕ (where m can range from 1 to k, the length of π_ϕ), given all observed actions from the execution of π_ϕ, α has to infer the ϕ agent goals (from $\mathcal{G}_\phi$) and generate a solution to a counterplanning task, namely a counterplan.

Definition 8 (Counterplan) *Given ϕ agent plan $\pi_\phi = (a_{m+1}, \dots, a_k)$, a plan $\pi_\alpha = (a_1, \dots, a_n)$ is a valid counterplan for $\pi_\phi = (a_{m+1}, \dots, a_k)$ if the joint execution of π_α and π_ϕ does not allow ϕ to achieve the goals in G_ϕ; $G_\phi \not\subseteq \Gamma_{\phi,\alpha}(s, \pi_\phi, \pi_\alpha)$.*

Our approach to solve counterplanning tasks assumes that α can delete (or add in the case of negated literals) some proposition that ϕ *needs* in order to achieve its goals. There could be different definitions for *needed literals*. We use planning landmarks in this work. Therefore, we impose two constraints: the seeking agent ϕ and the preventing agent α share some propositions, $F_\phi \cap F_\alpha \neq \emptyset$; and at least one action a in α model, $a \in A_\alpha$, must delete (add) at least one of ϕ's plan landmarks.

Algorithm 1 shows the high-level algorithm used to solve a counterplanning task from the perspective of α. The algorithm first solves a goal recognition problem using RECOGNIZEGOALS given a planning domain, initial conditions, a set of candidate goals $\mathcal{G}_\phi$, and a set of observations O_ϕ. It returns T_ϕ, a probability distribution over the set of candidate goals set $\mathcal{G}_\phi$ in the form of tuples $\langle$goal, probability$\rangle$. Then, the initial state of ϕ, I_ϕ, is updated with the given observations by advancing the state from the initial I_ϕ and applying all actions corresponding to the observations in O_ϕ. Next, we select the set of most probable goals' sets $G\prime_\phi$ from T_ϕ. For each goal $g \in G\prime_\phi$, we extract the landmarks of the new ϕ planning task using EXTRACTLANDMARKS. This computation will return the set of common landmarks among all the most probable sets of goals, $\mathcal{L}_{\Pi_\phi}$. Figure 1 shows an example of that computation in a navigation domain. If there are not common landmarks, the counterplanning task cannot be performed. Otherwise, the algorithm selects the set of counterplanning landmarks $\mathcal{L}_{\Pi_\phi,\Pi_\alpha}$ in EXTRACTCPLANDMARKS. This process will be explained in detail later. As before, if there are not counterplanning landmarks, the counterplanning task cannot be performed. Otherwise, one of the landmarks in $\mathcal{L}_{\Pi_\phi,\Pi_\alpha}$ is negated and returned as the preventing agent's goal G_α in SELECTGOAL. Finally, a plan π_α is computed to achieve that goal such that it prevents ϕ from achieving its goals. In the next section we discuss how we select G_α from $\mathcal{L}_{\Pi_\phi}$.

Algorithm 1 DOMAIN-INDEPENDENT COUNTERPLANNING

```
Inputs: $\Pi_\phi, \Pi_\alpha, \mathcal{G}_\phi, O_\phi$
Outputs: $\pi_\alpha$
 1: $T_\phi \leftarrow$ RECOGNIZEGOALS$(F_\phi, A_\phi, I_\phi, \mathcal{G}_\phi, O_\phi)$
 2: $I_\phi \leftarrow$UPDATE$(I_\phi, A_\phi, O_\phi)$
 3: $\mathcal{L}_{\Pi_\phi} \leftarrow F_\phi$
 4: $G\prime_\phi \leftarrow$ goal$(\arg\max_{t \in T_\phi}$ probability$(t))$
 5: $\pi_\alpha \leftarrow \emptyset$
 6: for $g \in G\prime_\phi$ do
 7:     $\mathcal{L}_{\Pi_\phi} \leftarrow \mathcal{L}_{\Pi_\phi} \cap$ EXTRACTLANDMARKS$(F_\phi, A_\phi, I_\phi, g)$
 8: if $\mathcal{L}_{\Pi_\phi} \neq \emptyset$ then
 9:     $\mathcal{L}_{\Pi_\phi,\Pi_\alpha} \leftarrow$EXTRACTCPLANDMARKS$(\Pi_\phi, \Pi_\alpha, \mathcal{L}_{\Pi_\phi})$
10:     if $\mathcal{L}_{\Pi_\phi,\Pi_\alpha} \neq \emptyset$ then
11:         $G_\alpha \leftarrow$SELECTGOAL$(\Pi_\phi, \Pi_\alpha, \mathcal{L}_{\Pi_\phi,\Pi_\alpha})$
12:         $I_\alpha$ =UPDATE$(I_\alpha, A_\alpha, O_\phi)$
13:         $\pi_\alpha \leftarrow$ PLANNER$(\Pi_\alpha = (F_\alpha, A_\alpha, I_\alpha, G_\alpha))$
14: return $\pi_\alpha$
```

3.1 Selecting Goals from Landmarks

Given a set of common landmarks L_{Π_ϕ}, two questions arises: (1) how many of those fact landmarks could be deleted (added) by α's model of the world (domain), so ϕ cannot achieve them?; and (2) within those facts that α can delete (add), which one should it become its goal G_α to effectively stop ϕ from achieving its goal? The first question brings us to the following definition:

Definition 9 Counterplanning landmark *Given the set of fact landmarks from Π_ϕ, $\mathcal{L}_{\Pi_\phi}$, a landmark $l_i \in \mathcal{L}_{\Pi_\phi}$ is a counterplanning landmark for α if $\exists a \in A_\alpha$ with $l_i \in$ eff(a). If l_i is a positive literal, l_i should be in $del(a)$. If l_i is a negative literal, l_i should be in $add(a)$.*

All the fact landmarks that comply with this condition are added to the counterplanning landmarks set of

[2]We have changed the notation o_i for a_j in the π_ϕ plan to differentiate between observations and future actions.

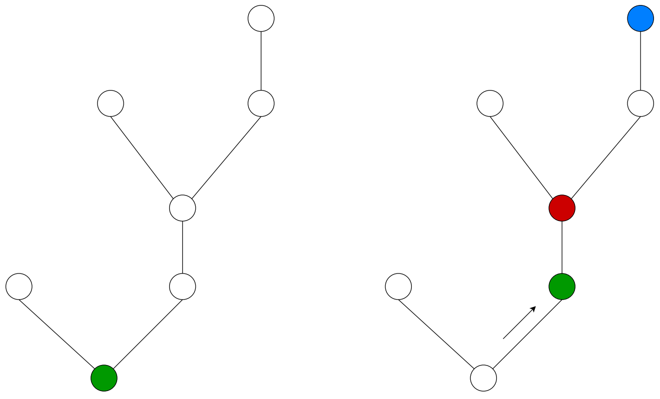

Figure 1: Navigation example. The green node illustrates the current position of the agent. The rest of the nodes represent a possible goal. The red node indicates a common landmark among the remaining goals after observing an action (moving to the right). The blue node refers to the actual goal of the seeking agent.

both agents $\mathcal{L}_{\Pi_\phi,\Pi_\alpha}$. We will refer to this process as EXTRACTCPLANDMARKS($\Pi_\phi, \Pi_\alpha, \mathcal{L}_{\Pi_\phi}$).

The second question requires further analysis. Given the definition of a fact landmark, α only has to delete (add) a fact landmark of Π_ϕ to prevent the seeking agent from achieving its goal. The problem therefore turns into selecting a single goal from $\mathcal{L}_{\Pi_\phi,\Pi_\alpha}$ to become the next goal for α, G_α. In most counterplanning domains, the earlier we discover the opponent's intentions (and thus stop him/her from achieving its goal), the better. This is a common characteristic in: (1) cyber security domains where we want to detect an intruder as soon as possible; (2) real-time strategy games where we want to defeat our enemy in the shortest possible time; or (3) a medical domain where we want to prevent the disease from spreading at the earliest time. Thus, this type of problems presents some temporal aspects that we need to take into account. In particular, it is not useful for α to pursue a counterplanning fact landmark that ϕ is going to achieve before α can avoid it. We define this temporal subproblem as finding the *First Counterplanning Landmark*, FCL. Algorithm 2 shows the high-level algorithm used to find it. For each counterplanning landmark l_i, an optimal plan is computed for ϕ and α. It is done optimally to ensure that the returned values correspond to the shortest time (cost) when both agents could reach that subgoal.

If the cost (duration) of achieving $\neg l_i$ by α solving Π_α is smaller than the cost (duration) of achieving l_i by ϕ, solving Π_ϕ, and there is no other landmark l_j with smaller cost, then $\neg l_i$ becomes FCL. In other words, l_i is the first landmark in Π_ϕ that α can achieve before ϕ. Therefore, the new G_α will be the negated FCL, since we want α to avoid ϕ achieving FCL. As a reminder, we are performing a one-step counterplanning episode. If ϕ performs some actions to re-achieve FCL or change its goals, then we assume α would have to start a new counterplanning episode.

3.2 Example

To illustrate our approach, let us consider a simple domain where a terrorist has committed an attack in the center of a city. Figure 2 shows the road network for this problem. The city police (α) knows that the terrorist (ϕ) wants to leave the city by either $G1$ (airport), $G2$ (train station), or $G3$ (bus terminal); so, $\mathcal{G}_\phi = \{G1, G2, G3\}$. The police has control over some cameras located at key points around the city (represented as nodes in Figure 2). The police actions consist of stopping the terrorist by setting a control at any of those points (A_α). So, the police wants to: (1) quickly know where the terrorist wants to go ($G\prime_\phi$); and (2) stop him/her as soon as possible to avoid panic breaking out. When the cameras observe that the terrorist is at L1 ($O_1 = a_1 \in A_\phi$), the police guesses that his/her goal is to reach $G1$ ($G\prime_\phi$) by doing goal recognition. The police only has resources to set one control. It knows that the terrorist must pass through L1, L2 and L3 to reach the airport. Although these four spots are counterplanning landmarks $\mathcal{L}_{\Pi_\phi,\Pi_\alpha}$, the police can only set the control at L2, L3 and $G1$ before the terrorist reaches those places. Finally, the police goal G_α will be to set the control at L2 since it is the FCL; i.e. the first spot where the terrorist can be effectively stopped.

Algorithm 2 SELECT GOAL

```
Inputs: $\Pi_\phi, \Pi_\alpha, \mathcal{L}_{\Pi_\phi,\Pi_\alpha}$
Outputs: FCL
 1: FCL $\leftarrow \emptyset$
 2: FCLCost $\leftarrow 0$
 3: for $l_i$ in $\mathcal{L}_{\Pi_\phi,\Pi_\alpha}$ do
 4:     $\pi_\phi \leftarrow$ PLANNER($\Pi_\phi = \langle F_\phi, A_\phi, I_\phi, l_i \rangle$)
 5:     $\pi_\alpha \leftarrow$ PLANNER($\Pi_\alpha = \langle F_\alpha, A_\alpha, I_\alpha, \{\neg l_i\} \rangle$)
 6:     if $c(\pi_\phi) >= c(\pi_\alpha)$ then
 7:         if $c(\pi_\phi) <$ FCLCost then
 8:             FCLCost $\leftarrow c(\pi_\phi)$
 9:             FCL$\leftarrow \neg l_i$
10: return FCL
```

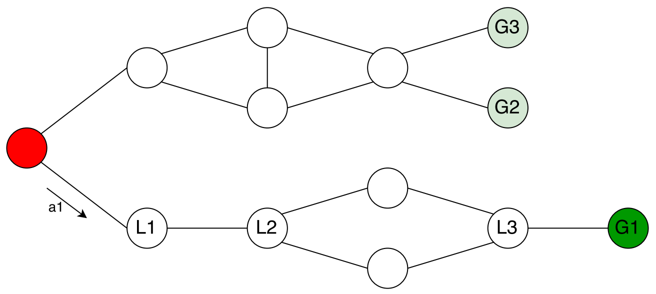

Figure 2: Terrorist capture domain. The red circle represents the initial position of the terrorist. Green circles represent his/her possible goals. Arrow a1 indicates the terrorist first observed action. L1, L2 and L3 refer to the landmark points that the terrorist has to pass through in order to reach $G1$.

4 Experiments and Evaluation

We empirically evaluate our approach on the new previously described TERRORIST domain as well as in other domains usually used in goal recognition works such as LOGISTICS, EASY IPC GRID, BLOCKS, and INTRUSION DETECTION. Each domain and problem conforms Π_ϕ. Additionally, in order to perform counterplanning, for each domain we have generated a new counterplanning domain, which defines the planning task Π_α for α. The classical domain and the counterplanning domain comply with the requirements mentioned in Section 3.

For each classical domain, we generate 10 random problems for ϕ. All the problems have the same number of objects. However, each problem has a different actual goal for ϕ, which is hidden for α. Their details are explained below.

- LOGISTICS. ϕ can deliver any package to any destination. It can do it by driving either trucks or planes. α can break a truck or a plane to interrupt the delivery.
- TERRORIST. ϕ may want to get to any point in the map. It can do it by navigating through points that are connected. α can set a control at a point so that they can arrest ϕ.
- INTRUSION DETECTION. ϕ may want to perform a set of attacks to a pool of computers. It can do it by performing hacking actions like delete logs or gain root access. α can perform administration actions like encrypt the information or change the root password to a computer to prevent ϕ from conducting its attack.
- BLOCKS. ϕ may want to put any of block on top of another. It can do it by picking up and stacking blocks that are not painted. α can paint a block so ϕ can not pick it up and preventing it from achieving its goal.
- EASY IPC GRID. ϕ may want to get to any cell in the grid. It can do it by navigating through cells without door or, if it has the right key, by opening them. α can navigate through those cells without needing a key, and can steal the key or change the lock of a door to block ϕ's plan.

The set of candidate goals $\mathcal{G}_\phi$ always consists of a 20% of all the possible goals in each problem. Therefore, bigger $\mathcal{G}_\phi$ sets for the same domain mean bigger problems. In particular, for the TERRORIST domain with a problem map of 20 nodes, $\mathcal{G}_\phi$ consists of 4 random destinations; when the map has 50 nodes, $\mathcal{G}_\phi$ consists of 10 possible destinations. The hidden goal, i.e., the current goal of ϕ, G_ϕ, that is unknown to α, is always on $\mathcal{G}_\phi$.

The set of observed actions was taken to be a subset of the plan solution π_ϕ, ranging from 10% of the actions, up to 70% of the actions. We did not include tests where the observed sequence is higher than 70% because our counterplanning approach degrades rapidly. The reason for this is that the number of counterplanning landmarks decreases as the number of observed actions increases.

Our fully automatic domain-independent counterplanning approach works with any combination of goal recognition and classical planning approaches. For purposes of these tests, we have selected the following configuration of goal recognition techniques and planners. For the goal recognition part of our counterplanning technique, we have tested the aforementioned domains and problems using the Ramírez and Geffner [2010] approach with different planners. The planners are HSP$^*_f$ [Haslum, 2008], an optimal planner; and LAMA [Richter *et al.*, 2011], a satisficing planner. LAMA is used in two modes: as a *greedy* planner that stops after the first plan is found GREEDY LAMA; and as an *anytime* planner that reports the best plan found in a given time window LAMA. The planning times for all the planners were set to 1800 seconds. In all the domains π_ϕ is computed using GREEDY LAMA. For optimal plan computations of FCL, we use HSP$^*_f$. All the experiments were ran on a Ubuntu machine with Intel Core 2 Quad Q8400 running at 2.66 GHz.

Table 1 summarizes the experimental results. For each planner, each row shows average performance over the 10 problems in each domain. Each column represents different measures of quality and performance:

- $|\mathcal{G}_\phi|$: number of goals in the candidate goal's set.
- $|\pi_\phi|$, $|\pi_\alpha|$: average plan length cost for each agent.
- $|\mathcal{L}_{\Pi_\phi}|$: number of landmarks of the seeking agent planning task.
- %Obs: percentage of the actions of π_ϕ in O_ϕ. Higher percentages of observations mean that more actions of ϕ's plan have already been observed by α and, thus, executed by ϕ.
- Q: fraction of times that the actual goal G_ϕ was found to be the most likely goal $G\prime_\phi$. In our experiments, if $G\prime_\phi$ consist of more than one goal, we select the one with more counterplanning landmarks as the most likely goal. Ideally, $Q = 1$.
- Q_t: average time in seconds taken for solving the goal recognition problems.
- E: fraction of times that α executing π_α succeeds in stopping ϕ in achieving its goals. Ideally, $E = 1$.
- Pe: penalty value computed as the number of steps in π_ϕ that are successfully performed divided by the length of π_ϕ. This penalty value represents the cost paid by π_α at each time step that has not stopped π_ϕ. Lower values of Pe indicate better performance, ideally $Pe = 0$.

As we can see in all the domains, the higher percentage of observations, the higher Q values, as expected. The goal recognition task becomes easier as more actions have been observed (as reported in other goal recognition works). Regarding E, the fraction of times that α blocks ϕ achieving its goals is clearly related to Q. Guessing the opponent's goal right usually involves more opportunities to block it. This is the case of INTRUSION DETECTION, where our counterplanning approach performs well. However, there are some cases in which we can badly guess ϕ's goal and still block its goal achievement (Q value is lower than E). This happens when our analysis of the goal recognition process identifies a common landmark (to stop ϕ's plan), but selects a wrong goal as in some BLOCKS instances.

The value of E is also closely related to the percentage of observations. Lower percentage values allow α to find many landmarks where to effectively block ϕ. On the other hand, if most of the actions in π_ϕ have already been observed (i.e. executed by the seeking agent), there will be just a few counterplanning landmarks to prevent ϕ from achieving its goal. This is also connected with the penalty values Pe. Lower percentage of observations imply that, if the opponent can be

						HSP*$_f$				LAMA				Greedy LAMA			
Domain	$\|\mathcal{G}_\phi\|$	$\|\pi_\phi\|$	$\|\pi_\alpha\|$	$\|\mathcal{L}_{\Pi_\phi}\|$	%Obs	Q	Q_t	E	Pe	Q	Qt	E	Pe	Q	Qt	E	Pe
LOGISTICS	6	9.4	1	12.3	10	0.6	4.2	0.7	0.3	0.6	5.0	0.7	0.3	0.6	2.6	0.6	0.2
					30	0.6	7.5	0.7	0.4	0.6	10.7	0.7	0.4	0.7	2.6	0.8	0.4
					50	0.7	10.0	0.8	0.5	0.7	16.3	0.8	0.5	0.7	2.7	0.8	0.5
					70	0.9	14.3	0.6	0.8	0.9	28.8	0.6	0.8	0.9	2.8	0.7	0.8
TERRORIST	4	3.8	1	3.8	10	0.5	3.8	0.6	0.5	0.6	2.8	0.6	0.5	0.6	2.6	0.6	0.5
					30	0.6	4.0	0.8	0.7	0.6	2.9	0.8	0.7	0.6	2.8	0.8	0.7
					50	0.6	5.0	0.8	0.8	0.6	3.0	0.8	0.8	0.6	2.9	0.8	0.8
					70	0.9	5.1	0.9	1.0	0.9	3.3	0.9	1.0	0.9	3.0	0.9	1.0
	10	5.6	1	4.1	10	0.3	283.2	0.5	0.5	0.3	211.4	0.5	0.5	0.3	210.4	0.5	0.5
					30	0.5	287.2	0.6	0.8	0.4	235.0	0.6	0.8	0.4	227.3	0.6	0.8
					50	0.6	307.1	0.4	0.8	0.6	248.6	0.4	0.8	0.6	269.6	0.4	0.8
					70	0.6	325.8	0.4	1.0	0.6	321.1	0.4	1.0	0.6	322.7	0.4	1.0
INTRUSION DETECTION	6	4.3	1	4.8	10	1.0	0.5	1.0	0.4	1.0	1.0	1.0	0.4	1.0	0.7	1.0	0.4
					30	1.0	0.4	1.0	0.7	1.0	1.0	1.0	0.7	1.0	0.7	1.0	0.7
					50	1.0	0.4	1.0	0.7	1.0	1.0	1.0	0.7	1.0	1.0	1.0	0.7
					70	1.0	0.3	1.0	0.9	1.0	0.8	1.0	0.9	1.0	1.0	1.0	0.9
BLOCKS	10	8.6	1	17.6	10	0.2	836.6	0.4	0.2	0.2	482.1	0.4	0.2	0.2	155.3	0.4	0.2
					30	0.3	910.7	0.6	0.5	0.3	531.9	0.6	0.5	0.3	175.3	0.6	0.5
					50	0.5	980.3	0.8	0.5	0.5	602.7	0.8	0.5	0.6	195.8	0.9	0.6
					70	0.8	1070.3	0.7	0.8	0.8	655.3	0.7	0.8	0.8	207.8	0.7	0.7
EASY IPC GRID	4	12.6	4.5	6.3	10	0.3	5.6	0.1	1.0	0.3	1.9	0.1	1.0	0.1	1.7	0.1	0.8
					30	0.3	7.3	0.3	0.6	0.3	2.0	0.3	0.6	0.3	2.0	0.4	0.78
					50	0.1	9.8	0.3	0.7	0.1	2.8	0.3	0.7	0.1	2.5	0.4	0.8
					70	0.3	15.3	0.0	∞	0.3	3.5	0.0	∞	0.3	3.1	0.0	∞

Table 1: Comparison of the counterplanning approach in five domains using optimal and approximated goal recognition methods. Figures shown are all averages over the set of problems as explained in the text. The metrics measured are: size of $\mathcal{G}_\phi$, length of plans for each agent, number of landmarks $|\mathcal{L}_{\Pi_\phi}|$, percentage of observations, goal recognition accuracy Q and its time Q_t, counterplanning accuracy E and penalty value Pe.

blocked, it could be done farther from the goal than if 50% or more of the plan has already been observed.

The number of landmarks of ϕ's planning task affects the counterplanning results. Domains with a higher number of landmarks will have a higher number of potential counterplanning landmarks where to block the opponent. The experiments confirm this aspect: domains such as LOGISTICS and BLOCKS are more likely to have more landmarks and the penalty values are smaller than in the other domains which just have a few landmarks.

Regarding planners' performance, GREEDY LAMA seems to achieve the best overall results both in terms of quality (Q, E, Pe) and time (Q_t). Since LAMA is an anytime planner, sometimes it takes more time to complete the goal recognition process than the optimal planner HSP*$_f$. However, all planners scale poorly to bigger problem instances where $\mathcal{G}_\phi$ increases. This entails worse goal recognition performance Q and, therefore, worse counterplanning performance E and P. We could speed-up our technique by computing plan cost estimations instead of actual plans in order to improve the performance. That would allow its use in real-time environments.

Summarizing, the best scenario for our counterplanning technique (high E values and low Pe values) would be when the preventing agent guesses the seeker's actual goal ($Q = 1$) with a low percentage of observed actions (very soon) and there is a high number of landmarks in the seeker's planning task which the preventing agent can delete (add).

5 Conclusions and Future Work

We have presented a novel fully automatic domain-independent approach for counterplanning, which is based on classical planning techniques. We have formally defined the counterplanning task involving two planning agents: an agent that seeks to achieve some goals; and an agent that tries to prevent its opponent from achieving its goals. To successfully block an agent in a domain-independent way, we: (1) recognize the opponent's goals by observing its performed actions; (2) identify the counterplanning landmarks of its planning task; and (3) generate a sequence of actions to block its goal achievement process as soon as possible. Results show the benefits of our approach on preventing the opponent from achieving its goals in several domains. Its performance depends on the ability of the preventing agent to quickly infer the hidden goal, and the number of landmarks of the seeker's planning task.

In this work, we assume we are given the candidate goals for the goal recognition process (as in the usual literature on goal recognition). Future work would consist on relaxing this assumption and consider F_ϕ as $\mathcal{G}_\phi$. We also assume unit action costs. In future work, we would generalize our approach by considering non-unit action costs. Additionally, a natural extension to this work would be to assume a seeking agent capable of changing his/her plans and goals. It seems to be also possible to extend our approach to deal with noisy observations and uncertainty on the seeking agent's behavior.

Acknowledgements

This work has been partially supported by MINECO projects TIN2014-55637-C2-1-R and TIN2017-88476-C2-2-R.

References

[Albrecht *et al.*, 1997] David W. Albrecht, Ingrid Zukerman, Ann E. Nicholson, and Ariel Bud. Towards a bayesian model for keyhole plan recognition in large domains. In *User Modeling*, pages 365–376. Springer, 1997.

[Boddy *et al.*, 2005] Mark S. Boddy, Johnathan Gohde, Thomas Haigh, and Steven A. Harp. Course of action generation for cyber security using classical planning. In *ICAPS*, 2005.

[Borck *et al.*, 2015] Hayley Borck, Justin Karneeb, Ron Alford, and David W. Aha. Case-based behavior recognition in beyond visual range air combat. In *FLAIRS*, 2015.

[Carbonell, 1978] Jaime G. Carbonell. Politics: Automated ideological reasoning. *Cognitive Science*, 2(1):27–51, 1978.

[Carbonell, 1981] Jaime G. Carbonell. Counterplanning: A strategy-based model of adversary planning in real-world situations. *Artificial Intelligence*, 16(3):295–329, 1981.

[Cox, 2007] Michael T. Cox. Perpetual self-aware cognitive agents. *AI magazine*, 28(1):32, 2007.

[E-Martín *et al.*, 2015] Yolanda E-Martín, María D. R-Moreno, and David E Smith. A fast goal recognition technique based on Interaction estimates. In *IJCAI*, 2015.

[Edelkamp *et al.*, 2009] Stefan Edelkamp, Carsten Elfers, Mirko Horstmann, Marcus-Sebastian Schröder, Karsten Sohr, and Thomas Wagner. Early warning and intrusion detection based on combined AI methods. In *ICAPS*, 2009.

[Geib and Goldman, 2009] Christopher W. Geib and Robert P. Goldman. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, 173(11):1101–1132, 2009.

[Ghallab *et al.*, 2004] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

[Haslum, 2008] Patrik Haslum. Additive and reversed relaxed reachability heuristics revisited. *International Planning Competition*, 2008.

[Hoffmann *et al.*, 2004] Jörg Hoffmann, Julie Porteous, and Laura Sebastia. Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, 22:215–278, 2004.

[Hoffmann, 2015] Jörg Hoffmann. Simulated penetration testing: From" dijkstra" to" turing test++". In *ICAPS*, 2015.

[Jarvis *et al.*, 2005] Peter A. Jarvis, Teresa F. Lunt, and Karen L. Myers. Identifying terrorist activity with ai plan recognition technology. *AI Magazine*, 26(3):73, 2005.

[Kabanza *et al.*, 2010] Froduald Kabanza, Philipe Bellefeuille, Francis Bisson, Abder Rezak Benaskeur, and Hengameh Irandoust. Opponent behaviour recognition for real-time strategy games. In *AAAI Workshop on Plan, Activity, and Intent Recognition*, 2010.

[Molineaux *et al.*, 2010] Matthew Molineaux, Matthew Klenk, and David W. Aha. Goal-driven autonomy in a navy strategy simulation. In *AAAI*, 2010.

[Obes *et al.*, 2013] Jorge Lucangeli Obes, Carlos Sarraute, and Gerardo Richarte. Attack planning in the real world. *arXiv preprint arXiv:1306.4044*, 2013.

[Ontañón *et al.*, 2013] Santiago Ontañón, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill, and Mike Preuss. A survey of real-time strategy game ai research and competition in starcraft. *IEEE Transactions on Computational Intelligence and AI in games*, 5(4):293–311, 2013.

[Pattison and Long, 2010] David Pattison and Derek Long. Domain independent goal recognition. In *STAIRS*, 2010.

[Pereira *et al.*, 2017] Ramon Fraga Pereira, Nir Oren, and Felipe Meneguzzi. Landmark-based heuristics for goal recognition. In *AAAI*, 2017.

[Ramírez and Geffner, 2009] Miquel Ramírez and Hector Geffner. Plan recognition as planning. In *IJCAI*, 2009.

[Ramírez and Geffner, 2010] Miquel Ramírez and Hector Geffner. Probabilistic plan recognition using off-the-shelf classical planners. In *AAAI*, 2010.

[Ramírez and Geffner, 2011] Miquel Ramírez and Hector Geffner. Goal recognition over POMDPs: Inferring the intention of a POMDP agent. In *IJCAI*, 2011.

[Richter and Westphal, 2010] Silvia Richter and Matthias Westphal. The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, 39:127–177, 2010.

[Richter *et al.*, 2011] Silvia Richter, Matthias Westphal, and Malte Helmert. LAMA 2008 and 2011. In *International Planning Competition*, 2011.

[Rowe, 2003] Neil C. Rowe. Counterplanning deceptions to foil cyber-attack plans. In *IEEE Workshop on Systems, Man and Cybernetics Society*, 2003.

[Sarraute *et al.*, 2012] Carlos Sarraute, Olivier Buffet, and Jörg Hoffmann. POMDPs make better hackers: Accounting for uncertainty in penetration testing. In *AAAI*, 2012.

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

[Vered and Kaminka, 2017] Mor Vered and Gal A. Kaminka. Heuristic online goal recognition in continuous domains. In *IJCAI*, 2017.

[Weber *et al.*, 2010] Ben George Weber, Michael Mateas, and Arnav Jhala. Applying goal-driven autonomy to starcraft. In *AIIDE*, 2010.

[Willmott *et al.*, 2001] Steven Willmott, Julian Richardson, Alan Bundy, and John Levine. Applying adversarial planning techniques to Go. *Theoretical Computer Science*, 252(1-2):45–82, 2001.