The AAAI-17 Workshop on
Plan, Activity, and Intent Recognition
WS-17-13

Plan Recognition Design

Reuth Mirsky, Roni Stern, Ya'akov (Kobi) Gal, Meir Kalech
Department of Information Systems Engineering
Ben-Gurion University of the Negev, Israel
+972-50-378-1474, dekelr@bgu.ac.il

Abstract

Goal Recognition Design (GRD) is the problem of designing a domain in a way that will allow easy identification of agents' goals. This paper extends the original GRD problem to domains that are described using hierarchical plans (GRD-PL), and defines the Plan Recognition Design (PRD) problem which is the task of designing a domain using plan libraries in order to facilitate fast identification of an agent's plan. While GRD can help to explain faster *which* goal the agent is trying to achieve, PRD can help in faster understanding of *how* the agent is going to achieve its goal. Building on the GRD paradigm, we define for each of these two new problems (GRD-PL and PRD) a measure that quantifies the worst-case distinctiveness of a given planning domain. Then, we study the relation between these measures, showing that the worst case distinctiveness of GRD-PL is a lower bound to the worst case plan distinctiveness of PRD, and that they are equal under certain conditions. Methods for computing each of these measures are presented, and we evaluate these methods in three known hierarchical planning domains from the literature. Results show that in many cases, solving the simpler problem of GRD-PL provides an optimal solution for the PRD problem as well.

Introduction

Goal Recognition is the problem of inferring an agent's goal from observations given a domain description (Winikoff *et al.* 2002; Sukthankar *et al.* 2014b; Ramırez and Geffner 2009). Keren et al. (2014) define Goal Recognition Design (GRD) as the problem of building a domain in a way that will minimize the number of observations needed to recognize an agent's goal. They introduce a worst-case distinctiveness (wcd) measure that is an upper bound on the number of observations needed to solve the GRD problem for a given domain, and showed how to compute this measure in domains based on the STRIPS representations.

An alternative representation for a planning domain is plan libraries (Wiseman and Shieber 2014; Kabanza *et al.* 2013; Blaylock and Allen 2006) which describe agents' activities as hierarchies of basic and complex actions. Plan libraries provide a rich domain representation but require the GRD task to explicitly reason about the hierarchical structure during domain design. To this end, the paper makes the following contributions:

First, it extends the GRD problem to domains represented using plan libraries (GRD-PL). We adapt the wcd computation to reasoning about hierarchical plans, and how to change a given domain in a way that will minimize its wcd without restricting the agent from reaching its possible goals. In this case, the wcd is computed using a search tree which finds the longest sequence of actions that the acting agent can perform before revealing its plan.

Second, it defines the Plan Recognition Design (PRD) problem, which is the task of designing a domain using plan libraries in order to facilitate fast identification of an agent's plan. In this case, identifying an agent's plan requires to infer the complete hierarchy of activities the agent is doing, not just its goal. Consider the case where a football player needs to decide about the strategy of a teammate. The goal of the teammate can be to attack or to defend, but there are various ways to execute each strategy. In order to help, the player must know the complete strategy of the teammate rather than just the goal: it is insufficient to know that the teammate is going to attack the goal; the player needs to know from which direction the teammate plans to attack, whether there will be a pass in this attack and when. All this information is part of the plan in the teammate's mind, and the player needs to infer it from observing the teammate. This is called the Plan Recognition problem, which is more challenging than the equivalent Goal Recognition problem (Sukthankar *et al.* 2014a; Blaylock and Allen 2006).

The paper formally defines PRD as the problem of minimizing the number of actions required to observe before unambiguously identifying the agent's plan. While GRD can help to explain faster *which* goal the agent is trying to achieve, PRD can help in faster understanding of *how* the agent is going to achieve its goal, allowing a deeper understanding of its activities which is crucial in providing assistance. For example, a tutor can guide a student's interaction in an educational software after it understands the student's intended solution strategy.

The paper defines a new measure, called worst-case plan distinctiveness, $wcpd$, which is analogous to the wcd measure in the GRD setting, and presents the longest sequence of observations that are required to recognize the agent's plan

	Input	Output	Metric	Meaning of Metric
GRD	STRIPS domain A list of goals	STRIPS domain with less actions	wcd	How many observations we need to see before we know the goal of the agent?
GRD-PL	Grammar-based plan library A list of goals	Plan Library with less rules	wcd	How many observations we need to see before we know the goal of the agent?
PRD	Grammar-based plan library A list of goals	Plan Library with less rules	wcpd	How many observations we need to see before we know the full plan of the agent?

Table 1: The scope of the paper is GRD-PL and PRD.

given a domain. We provide means of computing and minimizing the $wcpd$ measure and formalize the relationship between wcd and $wcpd$. Specifically, we show that the wcd in the GRD-PL setting is a lower bound of the $wcpd$ measure in the PRD setting and define the conditions under which they are equal. In particular, in some cases, solving a GRD-PL problem, which is shown to be easier computationally, can provide the optimal solution for the PRD problem as well.

The third contribution of this paper is an empirical evaluation of both GRD-PL and PRD paradigms on three known domains from the literature. We show the value and runtime of computing the wcd and $wcpd$ in these domains. We show empirically that in two of the domains, the values of the wcd measure are always equal to the $wcpd$ measure. This demonstrates that in practice, many times solving GRD-PL does provide an optimal solution for the PRD problem as well.

Table 1 summarizes the contributions of this paper.

Related Work

In Keren et al. (2014), the notion of worst-case distinctiveness (wcd) was introduced as a measure that assesses the ease of performing goal recognition within an environment. The wcd of a problem is the maximal length of a prefix of an optimal path an agent may take within a system before it becomes clear at which goal it is aiming. The objective in GRD is then to minimize the wcd without constraining the actor from achieving all goals. Later works by Keren et al. improved these approaches to account for non-optimal agents (2015) and non-observable actions (2016a).

Wayllace et al. (2016) defined GRD with stochastic agent action outcomes and Son et al. (2015) presented an approach to solving GRD using Answer Set Programming (ASP). Another recent work extends GRD to discuss partially observable environments, where the observer has partial observability of the agent's actions, while the agent is assumed to have full observability of its environment (Keren *et al.* 2016b). All these works assume a domain representation based on STRIPS and were not applied to hierarchical plans.

Another possible representation which allows hierarchical plans is grammar-based plan-libraries. The problem of grammar-based domain design was known for a long time. For context-free grammars, the metric $LL(k)$ is used to describe how many tokens of look-ahead are needed for a left linear parser without backtracking (e.g. until all ambiguity can be solved) (Rosenkrantz and Stearns 1969; Aho *et al.* 1975). However, these works discuss parsers of context-free grammars for compilation, without actual design of the grammars. Moreover, these works were not applied in the context of goal and plan recognition, where grammars are not necessarily context-free due to possible interleaving of actions.

A recent technical report (Ramirez and Geffner 2016) suggested an approach to compile plan libraries into a STRIPS problem representation. However, the compilation process is not without cost - the outputted problems are significantly larger than the original plan libraries (e.g. a plan library with 85 nodes becomes a STRIPS problem with 800 actions. A plan library with 251 nodes becomes a STRIPS problem with over 10,000 actions). The original GRD compilation was presented for domains with 8 to 920 grounded actions, and did not always manage to finish the reduction process for the larger domains. Thus, using GRD-PL and PRD to optimize plan libraries is a faster process than compiling the domain to STRIPS and using GRD on the outputted domain. Moreover, it is sometimes the case that plan libraries are chosen due to their descriptive representation (e.g., to visualize the hierarchical nature of the agent's activities). In such cases, even if the compilation to STRIPS is feasible, there is still a preference to use plan libraries.

Definitions

We define a plan library in the standard way it is defined in the plan recognition literature (Geib and Goldman 2009; Kabanza *et al.* 2010; Mirsky and Gal 2016)

Definition 1 *A plan library is a tuple $L = \langle B, C, G, R\rangle$, where B is a finite set of basic actions, C is a finite set of complex actions, $G \subseteq C$ the possible goals and R is a set of rules of the form $c \rightarrow \tau \mid O$, where $c \in C$, τ is a string from $(B \cap C)^*$ and $O = \{(i,j) \mid c_i \prec c_j\}$ where c_i, c_j refer to the i-th and j-th actions in τ respectively.*

Intuitively, B represents all of the atomically observable actions an agent can execute, C represents more complex or abstract actions with G as the goals the agent can try to achieve and each $r \in R$ represents how a complex action from C can decompose to a sequence of other actions. For $c_i, c_j \in \tau$, we say that $c_i \prec c_j$ if there exists an ordering constraint $(i < j) \in O$.

Throughout the following sections, we will use a running example to exemplify the terminology:

B $= \{run, block, kick\}$.

C = **G** = $\{Defend, Attack\}$.

R is a set with the following rules:

- $Defend \rightarrow (run, block) \mid O = \{(1,2)\}$.
- $Attack \rightarrow (run, kick) \mid \phi$.

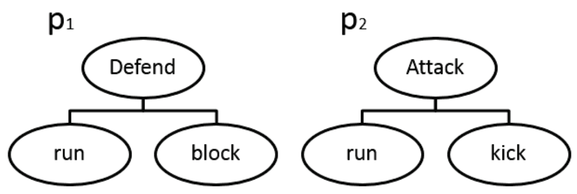

Figure 1: Two plans which can be constructed using the given plan library.

- $Attack \rightarrow (block, kick) \mid O = \{(1, 2)\}$.

This example is inspired by the soccer example used in several works from the plan recognition literature (Avrahami-Zilberbrand and Kaminka 2005; Tambe *et al.* 1999; Kitano *et al.* 1998), an agent can either execute a defense or an attack. The basic actions that can be observed are run, $block$ and $kick$. Notice that the rule for executing a defense plan has a constraint over the order by which the actions can be performed – the agent must first run and only then block, and not vice versa. There are also two rules which represent two different ways to perform an attack - one is by running and kicking (in either order) and the other is by blocking and then kicking the ball.

A plan is a labeled tree $p = (V, E, \mathcal{L})$, where V and E are the nodes and edges of the tree, respectively, representing the actions and their decomposition from the goal to the observable actions and $\mathcal{L}$ is a labeling function $\mathcal{L} : V \rightarrow B \cup C$ mapping every node in the tree to either a basic or a complex action in the plan library. The root note is called the goal of the plan and is labeled with a complex action from G. Each inner node is labeled with a complex action such that its children nodes are a decomposition of its complex action into constituent actions according to one of the rules. The set of all leaves of a plan p is denoted by $leaves(p)$. The notation $c_i \prec c_j$ is used to denote a pair of leaves c_i and c_j of a plan p where c_i must be performed before c_j. This occurs if there is such an ordering constraint in the rules used to create each node in the paths in p from the goal to c_i and c_j. Given a set of plans G, we define $Plans(g_i)$ to be the set of all plans in G with the root goal g_i and $Goal(p_i)$ to be the root goal of some plan p_i in G.

Figure 1 shows two plans. Plan p_1 is a plan which represents how a $Defend$ goal can be achieved – by performing two actions of $run, block$. Plan p_2 represents one way of how an $Attack$ goal can be achieved – by performing two actions of $run, kick$

An *observation sequence* is an ordered set of basic actions that represents actions carried out by the observed agent. A plan p *describes* an observation sequence O iff every observation is mapped to a leaf in the tree in an order that does not collide with the ordering constraints of the plan. More formally, a plan $p = (V, E, \mathcal{L})$ describes an observation sequence O iff there exists a partial function $f : O \rightarrow leaves(p) \cap B$ such that $\forall o \in O\ \exists v \in V\ f(o) = v$, and $\forall o_i, o_j \in O\ o_i < o_j \rightarrow \neg(f(o_j) \prec f(o_i))$. The last condition is needed to enforce the ordering constraints. Note that according to this definition, not all leaves of the plan are mapped to observations, meaning that a plan describes all observation sequences that can be mapped to its leaves and all of their prefixes. For each plan p we define $OBS(p)$ as the set of observation sequences such that each observations sequence in this set (1) is described by p, and (2) is not a prefix of a different observation sequence in $OBS(p)$. To illustrate, Figure 1 shows two plans based on the example plan library. Plan p_1 describes the observation sequence $\langle run, block \rangle$, the only sequence in $OBS(p_1)$. For the second plan $OBS(p_2) = \{\langle run, kick \rangle, \langle kick, run \rangle\}$ since in an attack plan, run and $kick$ can be performed in any order.

GRD with Plan Libraries (GRD-PL)

Keren et al. (2014) defined the wcd measure for any two goals g, g' as the longest sequence of observations required to observe before there is no ambiguity regarding which of the goals the acting agents is pursuing. The original GRD was defined using STRIPS for the domain description $P_D = \langle F, I, A \rangle$, where F are the fluents, $I \subseteq F$ is the initial state and A are the actions. For such a domain, $wcd(P) = max_{g,g' \in G} wcd(g, g')$.

Definition 2 *The Goal Recognition Design (GRD) problem is defined as a tuple $D = \langle P_D, G_D \rangle$ $\forall g \in G_D, g \subseteq F$ is a set of possible goals. The output of a GRD problem is $P'_D = \langle F, I, A' \rangle$ such that $A' \subseteq A$ and for all other domains P''_D that reduce the number of actions in A and the set of goals G_D, it holds that $wcd(P'_D) \leq wcd(P''_D)$ without restricting the agent from achieving any of the goals in G_D.*

We extend the GRD problem to deal with hierarchical plan libraries as follows:

Definition 3 *Goal Recognition Design (GRD-PL) is defined as a tuple $D = \langle P_D, G_D \rangle$, where P_D is a planning domain represented by a plan library $P_D = \langle B, C, G, R \rangle$ and G_D is a set of possible goals such that $G_D = G$. The output of a GRD-PL problem is $P'_D = \langle B, C, G, R' \rangle$ such that $R' \subseteq R$, $\forall P''_D \subseteq P_D\ \ wcd(P'_D) \leq wcd(P''_D)$, and every goal in G_D is achievable in P'_D.*

In our example plan library, the wcd between the goals in the plans in Figure 1 is 1. To illustrate, suppose that the first observation is run. In this case both plans p_1 and p_2 are still possible, so run cannot be used to distinguish between the goals $Attack$ and $Defend$. Any observation sequence of size 2 or more can be used to distinguish the goal of the agent. We therefore have $wcd(Attack, Defend) = 1$ and similarly wcd of the example PL is 1.

Finding the wcd in GRD-PL

The challenge of GRD-PL is in finding the wcd for a given domain. To this end, we propose to construct plans top-down for each goal until we reach a point where the observation sequences that can be described by each goal is unique. For a GRD-PL problem $\langle P_D, G_D \rangle$, we define a set of plans at iteration i,P_i.

P_0 is a set containing one plan per goal, where each plan has only one node, the root node, labeled with a unique goal

Algorithm 1: Calculating wcd

```
Input: $PL$: the complete plan library
Output: $wcd$: the length of the longest non-distinct
        sequence in $PL$
1  $wcd \leftarrow 0$
2  # $OPEN$ is the current set of plans
3  $OPEN \leftarrow \{g \mid g \in G\}$
4  # $Distinct$ is described in Algorithm 2
5  while $!Distinct(OPEN)$ do
6      $newOpen \leftarrow \emptyset$
7      $p \leftarrow OPEN.pop()$
8      $rules \leftarrow pl.rules()$
9      for $r \in rules\ s.t.\ r = c \rightarrow \tau | O$ do
10         if $c \in leftmost(p)$ then
11             $p' \leftarrow p.extend(c)$
               $newOpen \leftarrow newOpen \cap \{p'\}$
12     # To avoid losing completed plans
13     if $p.isComplete()$ then
14         $newOpen \leftarrow newOpen \cap \{p'\}$
15     $OPEN \leftarrow newOpen$
16 return $wcd$
```

from G_D. For each plan $p \in P_i$, we will create a plan $p' \in P_{i+1}$ for each possible rule that can extend p to p'.

To decide the set of possible rules that can be used to expand each plan, we use the *leftmost child* definition (Geib and Goldman 2009). For a node n representing a plan p, the node c is a leftmost child of p if:

- $c \in leaves(p)$.
- There exists a rule $r \in R$ such that $r = l \rightarrow \tau \mid O$ where $c \in \tau$.
- In O, there is no other child that precedes l and was not already expanded.

Given a plan p, we can expand p using a rule r iff the plan p has a leftmost child c and $r = c \rightarrow \tau \mid O$. This process guarantees a sound and complete process to expand the plan over possible rules (Geib and Goldman 2009).

The final iteration n is the one where the set of plans P_n is *distinctive*, meaning if it is possible to unambiguously recognize each goal in G_D from the observations that are described by the plans in P_n In this case we say that the goals in G_D are *distinct* from each other. Formally, $\forall g_1, g_2 \in G_D, g_1 \neq g_2 \rightarrow OBS(Plans(g_1)) \cap OBS(Plans(g_2)) = \emptyset$.

We now show how to extract the wcd of a plan library from the set of plans P_i. The wcd of the plan library is the size of the longest observation sequence described by one of the plans $p_i \in P_i$ that is also a prefix of another observation sequence described by $p_j \in P_i$ s.t. $Goal(p_i) \neq Goal(p_j)$. Algorithm 2 computes whether the goals in G_D are distinct and returns the wcd of P_D.

From (Keren *et al.* 2014), Corollary 1 proves that if an observations sequence O is distinct, then any observation sequence O' such that O is a perfix of O is distinct as well.

Algorithm 2: Check Goal Distinction

```
Input: $OPEN$: the plans in the current set of plans $P_i$.
Input: $G_D$: the set of goals in the plan library.
Output: $isDistinct$: whether the goals are distinct.
Output: $wcd$: the $wcd$ of the plan library.
1  $isDistinct \leftarrow True$
2  for $g_i \in G_D$ do
3      $P_i \leftarrow Plans(g_i) \cap OPEN$
4  for $p_i \in P_i, p_j \in P_j$ do
5      for $seq_i \in OBS(p_i), seq_j \in OBS(p_j)$ do
6          if $seq_i.contains(seq_j)$ then
7              $obsLength \leftarrow |seq_i|$
8              if $wcd < obsLength$ then
9                  $wcd \leftarrow obsLength$
10             $isDistinct \leftarrow False$
11 return $(isDistinct, wcd)$
```

This corollary can be used to enforce some improvements to the search process:

Distinctive goals if the set of plans $P_i^1 = P_i \cap plans(g_1)$ is distinct from the plans of all other goals, then we can stop expanding the plans in P_i^1. This means that if at some iteration i, there is no more ambiguity between the plans of a goal g_1 to the other goals, there will be no such ambiguity in the future as well and we can stop expanding this goal's plans.

Distinctive plans for some iteration i and a set of plans P_i, if a plan $p \in P_i$ is distinct from all other plans, when we can stop expanding p. This means that all $OBS(p)$ are distinct, and according to the Corollary, there is no sequence that can be added to $OBS(p)$ that will make it non-distinct, so we can stop expanding this plan.

We note that if the grammar contains recursive rules, we will need to set a recursion bound in order to make sure that the same rule won't be used infinitely. This is a common relaxation of plan libraries (Geib and Goldman 2009; 2011; Kabanza *et al.* 2013).

Reducing the wcd in GRD-PL

The search after a minimal set of rules to remove in order to minimize the wcd in GRD-PL can be done by traversing over all combinations of rules from the original plan library to find which combination provides the smallest wcd without restricting the acting agent from achieving each goal.

The stopping conditions are: (1) There are no more rules that can be removed, meaning all combinations that are removable (without hindering the ability of the acting agent to reach all goals) were tested; (2) We have reached a plan library with a wcd of 0.

Removing rules from the plan library will restrict the acting agent, and considering different combinations of rules can be costly. Therefore we propose an anytime search that gradually constrains the plan library, so that we first examine all removals of a single rule, then removals of two rules, etc. In the empirical section, we show that this anytime algorithm provides wcd which is close or equal to the optimal by reducing only 1 rule, in reasonable time.

Plan Recognition Design

Plan Recognition Design (PRD) is the problem of designing a domain in a way that will allow faster recognition of the plan of an acting agent. While in GRD the design tries to minimize the number of observations required until there is no ambiguity regarding the goal of the acting agent, PRD tries to minimize the number of observations required until there is no ambiguity regarding the complete plan of the acting agent. We define a new metric, *worst case plan distinctiveness (wcpd)*, which is defined as the number of observations we will need to see, in the worst case, until we know for certain what the agent's plan is. Formally, for each two plans p_1, p_2, we define

Definition 4 $wcpd(p_1, p_2) = \max_{O \in (OBS(p_1) \cap OBS(p_2))} |O|$

We extend this definition to plan libraries, such that the $wcpd$ of a plan library P_D is the maximal $wcpd$ of every pair of plans that can be generated for goals in P_D. Formally, $wcpd(P_D) = \max_{p_1, p_2 \in \bigcup_{g \in G} Plans(g)} wcpd(p_1, p_2)$.

The $wcpd$ of the plan library from our running example is 1, since after observing run there is still ambiguity regarding the plan of the agent. Note that an additional observation (e.g., $kick$) will unambiguously identify the plan of the agent. We are now ready to define the new problem of Plan Recognition Design, which aims to minimize the plan library's $wcpd$.

Definition 5 *Plan Recognition Design (PRD) is defined as a tuple $D = \langle P_D, L_D \rangle$, where P_D is a planning domain represented by a plan library $\langle B, C, G, R \rangle$ and L_D is a set of possible plans such that $L_D = \bigcup_{g \in G} Plans(g)$. The output of a PRD problem is $P'_D = \langle B, C, G, R' \rangle$ such that $R' \subseteq R$ and $\forall P''_D \subseteq P_D \;\; wcpd(P'_D) \leq wcpd(P''_D)$.*

Next, we study the relation between PRD and GRD, and compare the costs of computing them.

Lemma 1 *The wcd of any two different goals is the maximal $wcpd$ of the plans that achieve these goals:*
$wcd(g_i, g_j) = \max_{p_i \in Plans(g_i), p_j \in Plans(g_j)} wcpd(p_i, p_j)$.

Proof: Let $P_i = Plans(g_i)$ for $i = 1, 2$. Assume by contradiction to the lemma that there are plans p_1, p_2 such that $Goal(p_i) = g_i$ and $wcpd(p_1, p_2) > wcd(Goal(p_1), Goal(p_2))$. This means that there is an observations sequence of length higher than $wcd(Goal(p_1), Goal(p_2))$ that can be described both by a plan to achieve g_1 and a plan to achieve g_2,but this contradicts $wcd(g_1, g_2) = wcd(Goal(p_1), Goal(p_2))$. Thus, $wcd(g_1, g_2) = max_{p_1 \in Plans(g_1), p_1 \in Plans(g_1)} wcpd(p_1, p_1)$□

Theorem 1 *For every pair of plans p_i, p_j, if $Goal(p_i) \neq Goal(p_j)$, it holds that $wcpd(p_i, p_j) \leq wcd(Goal(p_i), Goal(p_j))$.*

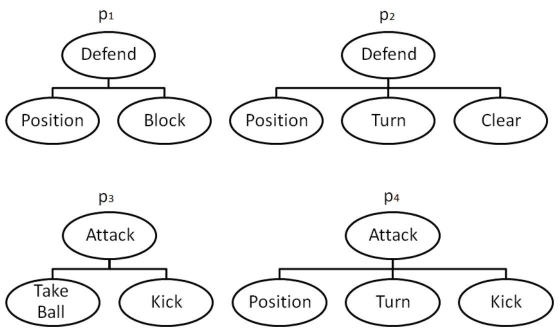

Figure 2: Plans for “Defend” and “Attack”.

Proof: Let p_1, p_2 be plans such that $p_1 \in Plans(g_1)$ and $p_2 \in Plans(g_2)$. If $g_1 \neq g_2$, then according to Lemma 1, $wcd(Goal(p_1), Goal(p_2)) = max_{p_i \in Plans(g_1), p_j \in Plans(g_2)} \geq wcpd(p_1, p_2)$. If $g_1 = g_2$, then $wcd(g_1, g_2) = 0 \leq wcpd(g_1, g_2)$□

To illustrate, we show that the $wcpd$ of two plans can be strictly smaller than the wcd of their respective goals. Such an example is presented in Figure 2, where we assume that the actions must be fully ordered. $Defend$ and $Attack$ are two goals from P_D, $Plans(Defend) = \{p_1, p_2\}$ and $Plans(Attack) = \{p_3, p_4\}$. Following Lemma 1 and Theorem 1 we get that:[-0.5em]

$$wcd(Goal(p_1), Goal(p_3)) = wcd(Defend, Attack) = max_{p_i \in Plans(Defend), p_j \in Plans(Attack)} wcpd(p_i, p_j) = 1 > wcpd(p_1, p_3) = 0 \quad (1)$$

The discussion so far was about the $wcpd$ of plans assuming that the goals they aim to achieve are different. However, when we discuss the measures of a complete plan library, we need to also consider the $wcpd$ of plans that have the same goal. In such cases, $wcpd(p_1, p_2)$ might be bigger than zero while $wcd(Goal(p_1), Goal(p_2)) = 0$.

Theorem 2 *For every plan library P_D it holds that $wcd(P_D) \leq wcpd(P_D)$.*

Proof: We prove this by separating all pair of plans into two groups: the first group $G_=$ holds all pairs p_1, p_2 such that $Goal(p_1) = Goal(p_2)$, and the second group $G_{\neq}$ contains all pairs of plans with different goals. According to Definition 4, $wcpd(P_D) = max_{p_1, p_2 \in L_D} wcpd(p_1, p_2) = max_{G \in G_=, G_{\neq}} wcpd(G)$. For $G_=$, we know that

$$wcpd(G_=) = max_{\langle p_1, p_2 \rangle \in G_=} wcpd(p_1, p_2) \geq wcd(Goal(p_1), Goal(p_2)) = wcd(G_=) = 0. \quad (2)$$

For G_{neq}, we know that

$$wcpd(G_{neq}) = max_{p_i, p_j} wcpd(p_i, p_j) = max_{p_k \in Plans(Goal(p_i)), p_l \in Plans(Goal(p_j))} wcpd(p_k, p_l) = max_{p_i, p_j \in G_2} wcd(Goal(p_i), Goal(p_j)) = wcd(G_{neq})\Box \quad (3)$$

Calculating the $wcpd$ of a domain can be performed using the same search tree of the wcd calculation. The only change will be the stopping condition – where each goal contains an exclusive set of plans. The $wcpd$ then will be the longest length of OBS of these plans. The algorithm for finding the *distinctiveness* of the current plans in the open list is described in Algorithm 2.

Reducing the $wcpd$ can also be performed using the same search tree of the wcd calculation. The change will only be in the stopping conditions – in this search, we will not stop even if the wcd is 0, since we might still be able to reduce the $wcpd$.

The cost of computing the $wcpd$ of a domain is higher than the cost of computing its wcd. There are two parts for this computation - first, the cost of expanding a level in the search tree; and second, the cost of deciding whether another level should be expanded.

The first level requires us to produce $|G|$ nodes, one node per possible goal. Expanding the search tree from the $i-1$th level to the ith level, requires us to create child nodes for all leftmost children from level $i-1$. For each node n describing plan p, and for every leftmost child c, we need $|r'|$ edges from n to its child nodes s.t. $r' \subseteq r \mid r' : c \rightarrow \tau \mid O$. Let τ_{max} be the rule with the longest righthand side, then if we have k nodes in level $i-1$, at the worst case we will have all possible leaves as leftmost children such that we will have $k \cdot (|\tau_{max}| - 1)^{i-1} \cdot |r|$ nodes in the ith level. In total, expanding k levels of the search tree will require us $O(|G| \cdot \Sigma_{i=1}^{k}(|\tau_{max}| - 1)^{i} \cdot |r|^{i-1})$.

Next, when we reach a level with k plans, we need to see if these nodes represent plans which describe a mutually exclusive sets of observations. Let $G_1, \ldots, G_n$ be the set of goals in the plan library, and let $P_1, \ldots, P_n$ be the number of different plans for each goal, i.e., P_i is the number of plans for goal G_i. Denoting X as the cost of checking the longest observation sequence described by a pair of plans (Definition 4), then the runtime of computing the longest observation sequence between different goals for a complete plan library, denoted T_{wcd}, is $T_{wcd} = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} P_i \cdot P_j \cdot X$.

Next, we need to combine these two results to show the complete calculation time of expanding the search tree, with the distinctiveness check at each level. Together we get a runtime of $O(|G| \cdot \Sigma_{i=1}^{k}(T_{wcd} \cdot |\tau_{max}|)^{i} \cdot |r|^{i-1})$.

Since the wcd of a plan library is smaller than or equal to the $wcpd$, this result highlights that computing the wcd can be done faster than computing the $wcpd$. This advantage is mainly present when $wcpd$ and wcd are very different. As we show in the next section, in practice it is sometimes enough to find a reduction to the wcd, which is easier to compute, in order to find a solution for the $wcpd$ reduction as well.

Empirical Evaluation

We evaluate our work in three common domains from the plan recognition literature. The first is the Monroe, a disaster management domain (Blaylock and Allen 2005). The second domain is the Soccer domain from Avrahami-Zilberbrand and Kaminka (2005), which was inspired by the behavior hierarchies of RoboCup soccer teams (Kitano *et al.* 1997; Tambe *et al.* 1999). The third is a simulated domain generated to evaluate plan recognition algorithms (Geib *et al.* 2008; Kabanza *et al.* 2013; Mirsky *et al.* 2016). The benefit of the simulated domain is that it allows different configurations to better study the algorithms.

	wcd	$wcpd$	wcd time (sec.)	$wcpd$ time (sec.)
Monroe	6	10	0.24	4.0
Soccer	2	2	0.06	0.069
Simulated1	0.82	0.82	7.37	7.33
Simulated2	3.66	3.89	9.71	177.79

Table 2: wcd and $wcpd$ values for all domains (for the simulated domains, the reported values are averages).

In Monroe, there are 30 basic actions, 40 basic actions, 10 possible goals and 49 rules. The rules are partially ordered, meaning that a sequence of actions can be executed in several orders.

The Soccer domain is relatively small in size and all actions are fully ordered. It has 7 basic actions, 10 complex actions, 3 possible goals and 13 rules.

We used two types of simulated domains, named Simulated1 and Simulated2. Both domains included 30 complex actions, 2 possible goals and 50 rules with partial ordering. Simulated1 had 100 basic actions, which was similar to the configuration used in the prior work mentioned above. Simulated2 had 20 basic actions (a smaller vocabulary) which made the domain more ambiguous (MacDonald *et al.* 1994), as there can be more possible plans for completing each goal.

The wcd and $wcpd$ values of these domains, and their computation time, are summarized in Table 2. The values for the simulated domains are averaged over the 100 generated instances. As shown in the table, the wcd and $wcpd$ varied across the different domains, and the value of wcd was smaller than or equal to the value of $wcpd$ for all domains, consistent with Theorem 2. In addition, the computation time for wcd was smaller than or equal to the computation time of $wcpd$.

In two domains, Soccer and Simulated1, the wcd and $wcpd$ values, as well as their computation time, were practically equal. This is consistent with the runtime analysis shown in Section 11.

In the Monroe domain, the value for the $wcpd$ was significantly higher than the value for wcd. We attribute this fact to the high number of possible plans for achieving the same goal. To illustrate, there is a possible plan for providing electricity which includes, or does not include, driving to get fuel for the generator.

The time required to compute the $wcpd$ for the Simulated2 domain was significantly higher than that for Simulated1, which we attributed to the higher ambiguity of this domain.

Next, we evaluate the ability to modify the plan library in the different domains in order to reduce the wcd and $wcpd$.

	1-Rule			All Rules		
	wcd	$wcpd$	Time	wcd	$wcpd$	Time
Monroe	0	0	7.31	–	–	–
Soccer	50%	50%	1.26	50%	50%	6.53
Simulated1	82%	82%	720	92%	92%	2,326
Simulated2	49%	51%	2,141	83%	84%	6,580

Table 3: wcd and $wcpd$ reduction.

Table 3 summarizes the percentage of reduction of the wcd and $wcpd$ values for the following conditions: removing a single rule, and removing all possible combinations of rules which represent the upper bound on the reduction potential of the domain. These results also demonstrate the anytime capabilities of the search algorithm.

As shown in the table, for the Soccer domain, we were able to reduce the wcd and $wcpd$ by 50% by removing a single rule from the plan library. This is the maximal possible reduction for this domain. For the Simulated1 domain, we were able to achieve 90% of the possible reduction potential for this domain. For the more ambiguous Simulated2 domain, we were able to realize 60% of the reduction potential for this domain. Interestingly, in the Monroe domain, the wcd and $wcpd$ values could not be reduced by removing one rule, as well as removing all combinations of two to four rules from the plan library (not shown in the table). There were 49 rules in this domain, hence considering the removal of all possible 2^{49} combinations was not feasible.

Conclusion

This paper presented two new approaches for plan- and goal-recognition domain design: Plan Recognition Design (PRD), is the task of designing a domain using plan libraries in order to facilitate fast identification of an agent's plan. Goal Recognition Design with Plan Libraries (GRD-PL) extends Goal Recognition Design (Keren *et al.* 2014) approach to using plan libraries as a domain representation.

For each of these approaches the paper defines a new measure – wcd and $wcpd$ respectively – and provides an anytime algorithm for computing them and optimizing the recognition domain accordingly. We show that the wcd of GRD-PL is a lower bound to the $wcpd$ for PRD, and are equal under certain conditions. Therefore, solving the (simpler) GRD-PL problem may at times yield an optimal solution for the PRD problem. We evaluated our approach in three known hierarchical planning domains from the literature using the new measures. Our empirical work provides a novel evaluation for known plan libraries as well as how plan libraries can be modified in the different domains in order to reduce their wcd and $wcpd$. We showed that in two out of three domains, the anytime algorithm was able to achieve a significant reduction in the wcd and $wcpd$ of the domains in reasonable time. These results demonstrate the applicability of domain design for plan and goal recognition.

We are currently extending our work in several directions. First, removing rules from plan libraries might put heavy restrictions on agent's actions. We are considering other, less restrictive, manipulations on the domain, such as adding ordering constraints on actions. We are also designing heuristics for ordering which rule combinations to consider for removal from the plan libraries.

References

A.V. Aho, S.C. Johnson, and J.D. Ullman. Deterministic parsing of ambiguous grammars. *Communications of the ACM*, 18(8):441–452, 1975.

D. Avrahami-Zilberbrand and G.A. Kaminka. Fast and complete symbolic plan recognition. In *International Joint Conference of Artificial Intelligence (IJCAI)*, volume 14, 2005.

N. Blaylock and J. Allen. Generating artificial corpora for plan recognition. In *International Conference on User Modeling*, pages 179–188. Springer, 2005.

N. Blaylock and J. Allen. Fast hierarchical goal schema recognition. In *National Conference on Artificial Intelligence*, volume 21, page 796. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

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

C.W. Geib and R.P. Goldman. Recognizing plans with loops represented in a lexicalized grammar. In *AAAI*, 2011.

C.W. Geib, J. Maraist, and R.P. Goldman. A new probabilistic plan recognition algorithm based on string rewriting. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 91–98, 2008.

F. Kabanza, P. Bellefeuille, F. Bisson, A.R. Benaskeur, and H. Irandoust. Opponent behaviour recognition for real-time strategy games. In *Plan, Activity, and Intent Recognition*, 2010.

F. Kabanza, J. Filion, A. R. Benaskeur, and H. Irandoust. Controlling the hypothesis space in probabilistic plan recognition. In *International Joint Conference of Artificial Intelligence (IJCAI)*, pages 2306–2312, 2013.

S. Keren, A. Gal, and E. Karpas. Goal recognition design. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2014.

S. Keren, A. Gal, and E. Karpas. Goal recognition design for non-optimal agents. In *AAAI*, pages 3298–3304, 2015.

S. Keren, A. Gal, and E. Karpas. Goal recognition design with non-observable actions. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

S. Keren, A. Gal, and E. Karpas. Privacy preserving plans in partially observable environments. International Joint Conference of Artificial Intelligence (IJCAI), 2016.

H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. Robocup: The robot world cup initiative. In *Proceedings of the first international conference on Autonomous agents*, pages 340–347. ACM, 1997.

H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. The robocup synthetic agent challenge 97. In *RoboCup-97: Robot Soccer World Cup I*, pages 62–73. Springer, 1998.

M.C. MacDonald, N.J. Pearlmutter, and M.S. Seidenberg. The lexical nature of syntactic ambiguity resolution. *Psychological review*, 101(4):676, 1994.

R. Mirsky and Y. Gal. Slim: Semi-lazy inference mechanism for plan recognition. In *International Joint Conference of Artificial Intelligence (IJCAI)*, 2016.

R. Mirsky, R. Stern, Y. Gal, and M. Kalech. Sequential plan recognition. In *International Joint Conference of Artificial Intelligence (IJCAI)*, 2016.

M. Ramırez and H. Geffner. Plan recognition as planning. In *AAAI*, 2009.

M. Ramirez and H. Geffner. Heuristics for planning, plan recognition and parsing. *arXiv preprint arXiv:1605.05807*, 2016.

D.J. Rosenkrantz and R.E. Stearns. Properties of deterministic top down grammars. In *Proceedings of the first annual ACM symposium on Theory of computing*, pages 165–180. ACM, 1969.

Tran Cao Son, Orkunt Sabuncu, Christian Schulz-Hanke, Torsten Schaub, and William Yeoh. Solving goal recognition design using asp. 2015.

G. Sukthankar, C.W. Geib, H.H. Bui, D. Pynadath, and R.P. Goldman. *Plan, activity, and intent recognition: theory and practice*. Newnes, 2014.

G. Sukthankar, R. P. Goldman, C. Geib, D. V Pynadath, and H. H. Bui. An introduction to plan, activity, and intent recognition. 2014.

M. Tambe, G.A. Kaminka, S. Marsella, I. Muslea, and T. Raines. Two fielded teams and two experts: A robocup challenge response from the trenches. In *International Joint Conference of Artificial Intelligence (IJCAI)*, pages 276–283, 1999.

C. Wayllace, P. Hou, W. Yeoh, and T.C. Son. Goal recognition design with stochastic agent action outcomes. 2016.

M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative and procedural goals in intelligent agent systems. In *International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufman, 2002.

S. Wiseman and S. Shieber. Discriminatively reranking abductive proofs for plan recognition. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2014.