Goal Recognition Design

Sarah Keren, Avigdor Gal
{sarahn@tx,avigal@ie}.technion.ac.il
Technion — Israel Institute of Technology

Erez Karpas
karpase@csail.mit.edu
Massachusetts Institute of Technology

Abstract

We propose a new problem we refer to as goal recognition design (*grd*), in which we take a domain theory and a set of goals and ask the following questions: to what extent do the actions performed by an agent within the model reveal its objective, and what is the best way to modify a model so that any agent acting in the model reveals its objective as early as possible. Our contribution is the introduction of a new measure we call *worst case distinctiveness* (*wcd*) with which we assess a *grd* model. The *wcd* represents 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. To model and solve the *grd* problem we choose to use the models and tools from the closely related field of automated planning. We present two methods for calculating the *wcd* of a *grd* model, one of which is based on a novel compilation to a *classical planning* problem. We then propose a way to reduce the *wcd* of a model by limiting the set of available actions an agent can perform and provide a method for calculating the optimal set of actions to be removed from the model. Our empirical evaluation shows the proposed solution to be effective in computing and minimizing *wcd*.

Introduction

Goal recognition is a subproblem of plan recognition with the objective of discovering the terminal goal of an agent given its behaviour (Pattison and Long 2011). We propose a new related problem that we refer to as *goal recognition design* (*grd*). A *grd* problem takes a domain theory and a set of goals and answers the following two questions: (1) to what extent do the actions performed by an agent within the model reveal its objective, and (2) what is the best way to modify a model so that any agent acting within it reveals its objective as early as possible.

Goal recognition design is relevant to any domain for which quickly performing goal recognition is essential and in which the model design can be controlled. Applications of goal recognition design may be found in many problems where goal recognition is useful, including intrusion detection (Jarvis, Lunt, and Myers 2004; Kaluza, Kaminka, and Tambe 2011; Boddy et al. 2005), assisted cognition (Kautz et al. 2003), and natural language processing (Geib and Steedman 2007). For example, due to security reasons, tracking the activity of passengers in an airport may be performed in order to detect where passengers are heading. In addition, it is possible to set up barriers that control the flow of the passengers to improve the goal recognition task, but equally important to minimize the obstruction to the ease of use of the airport.

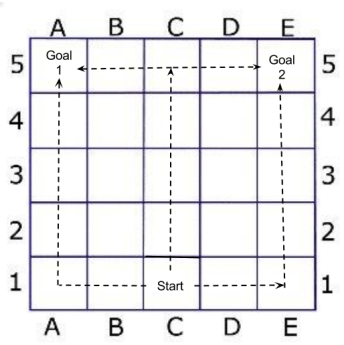
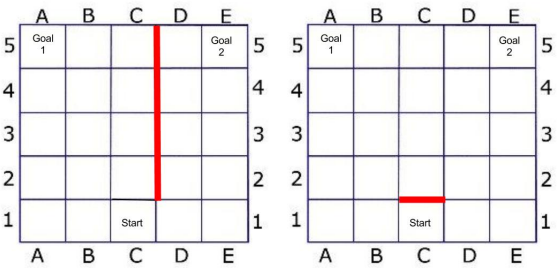


Figure 1: An example of a goal recognition design problem

As a first stage of our exploration of the new problem we assume agents are optimal and that the actions of the agent are fully observable. In order to achieve our objective, we introduce a new concept called *worst case distinctiveness* (*wcd*), which represents the maximal length of a prefix of a path an agent may take within a system before its objective becomes clear.

Figure 1 offers a simple example that will help clarify the concepts of our work. The model consists of a simple room (or airport) with a single entry point, marked as 'Start' and two possible exit points (boarding gates), marked as 'Goal 1' (domestic flights) and 'Goal 2' (international flights). An agent can move vertically or horizontally from 'Start' to one of the goals. Notice that for each of the goals there are several optimal paths, some of which share a common prefix with an optimal path to the other goal. In this model the goal of the agent becomes clear once turning left or right. Therefore, the *wcd* is 4 since in the worst case an optimal agent can move up 4 steps before it is obliged to turn towards its goal.

The *wcd* value helps in assessing a model and our goal is to reduce it. Towards this end, we define the means avail-



(a) A non-optimal solution (b) An optimal solution

Figure 2: Solutions for reducing the *wcd*

able for modifying a model. Since we want to limit the set of possible optimal paths of an agent, an elegant way of introducing changes into the model is by limiting the set of available actions an agent can perform. As a way of maintaining "user comfort" in the model we require the solution to preserve the original optimal solution length of all goals. In addition, given the maximal reduction achieved while respecting this constraint, we wish to minimize the change introduced to the model.

In our motivating airport example, the solution of placing barriers or screens in order to direct the flow of passengers is a common and effective solution. Figure 2 portrays two possible solutions that reduce the *wcd* of a model from 4 to 0. Clearly, the option in Figure 2(b) is preferable since it offers the same result by disallowing a single action and creating a single barrier.

Goal recognition design, while relevant to goal recognition — interchangeably called in the literature plan recognition (Kautz and Allen 1986; Cohen, Perrault, and Allen 1981; Lesh and Etzioni 1995; Ramirez and Geffner 2009; Agotnes 2010; Hong 2001) — is a totally different task. While goal recognition aims at discovering the goals (and sometimes plans) of an agent according to observations of its actions collected online, goal recognition design does not include a specification of an observation sequence, but rather offers an offline solution for assessing and minimizing the maximal number of observations that need to be collected in order to assure the goal of any optimal agent in the system is recognized.

To model and solve the *grd* problem we choose to use the models and tools from the closely related field of automated planning. The advantage of this choice is that by compiling our problem into a *classical planning* problem we gain the option of using various established tools and techniques.

The paper is organized as follows. We start by providing the necessary background on classical planning. We continue by introducing the formal model representing the *grd* problem and the *wcd* value. The following sections present the methods we developed for calculating and reducing the *wcd* value of a given *grd* problem. We conclude with an empirical evaluation, a discussion of related work, and a conclusion.

Background: Classical Planning

The basic form of automated planning, referred to as *classical planning*, is a model in which the actions of the agents within it are fully observable and deterministic. A common way to represent classical planning problems is the STRIPS formalism (Fikes and Nilsson 1972). A STRIPS planning problem is a tuple $P = \langle F, I, A, G, C \rangle$ where F is the set of fluents, $I \subseteq F$ is the initial state, $G \subseteq F$ represents the set of goal states, and A is a set of actions. Each action is a triple $a = \langle pre(a), add(a), del(a) \rangle$, that represents the precondition, add, and delete lists respectively, and are all subsets of F. An action a is applicable in state s if $pre(a) \subseteq s$. If action a is applied in state s, it results in the new state $s' = (s \setminus del(a)) \cup add(a)$. $C : A \to R^{0+}$ is a cost function on actions, that assigns each action a non-negative cost.

Given a planning problem, the objective is to find a plan $\pi = a_1, \ldots, a_n$, a sequence of actions that brings an agent from I to a state that satisfies the goal. The cost $c(\pi)$ of a plan π is $\Sigma(c(a_i))$. Often, the objective is to find an optimal solution for P, an optimal plan, π^*, that minimizes the cost. We assume the input of the problem includes actions with a uniform cost equal to 1. This means that plan cost is equivalent to plan length, and the optimal plans are the shortest ones.

Goal Recognition Design

We define a goal recognition design (*grd*) problem as a tuple $D = \langle P_D, \mathcal{G}_D \rangle$, where $P_D = \langle F, I, A \rangle$ is a planning domain formulated in STRIPS and $\mathcal{G}_D$ is a set of possible goals $G, G \subseteq F$ (whenever D is clear from context we will use P and G). It is worth noting that the model includes an initial state, common to all agents acting in the system. In case there are multiple initial states, there is a simple compilation, which adds a zero cost transition between a dummy common initial state and each initial state, making the model and the methods we propose applicable.

Given a *grd* problem, our objective is to find a measure that assesses the ability to perform goal recognition within the model and compute it. Once such a measure is established, we would like to modify the model such that the recognition ability is maximized, while respecting the specified constraints. There are several issues that need to be explained: first, we need to define a measure for the ability to recognize the objective of an agent in a concrete form and what it means to maximize it. Secondly, we need to make precise the allowed modifications to the model that have the potential of improving this ability. Before we elaborate on these two issues we detail three assumptions on which our model is based.

1. Agents in the system are acting optimally,
2. the outcomes of the actions are deterministic, and
3. the model is fully observable to the agent and system.

The removal of any combination of these assumptions creates an interesting scenario and a challenging problem to solve. However, as a first stage we want to keep our model as straightforward as possible and defer more complex problems to future work.

We start by defining what it means for the goal of the agent to be clear. As mentioned above, we are concerned with the design of a model and not with the behaviour of a specific agent. Accordingly, we need to recognize the options an agent may have within a model and examine the characteristics of the various paths available. Let $\Pi^*(G)$ represent the set of optimal paths to G.

Definition 1 *Given a problem $D = \langle P, \mathcal{G} \rangle$, a sequence of actions π is a* non-distinctive *path in D if $\exists G', G'' \in \mathcal{G}$ s.t. $G' \neq G''$ and $\exists \pi' \in \Pi^*(G')$ and $\pi'' \in \Pi^*(G'')$ s.t. π is a prefix of π' and π''. Otherwise, π is* distinctive.

We aim at creating a model in which agents reveal their goal as early as possible. The *wcd* value we propose serves as an upper bound on the number of actions any optimal agent can perform in a model before revealing its goal by selecting a distinctive path. Using the definition above, we can formally define the *wcd* value of a model as follows.

Definition 2 *Let $\Pi_D = \langle \pi | \pi$ is a non-distinctive path of $D \rangle$ and let $|\pi|$ denote the length of a path π. Then,* worst case distinctiveness *(*wcd*) of a model D, denoted by* wcd(D)*, is:*

$$wcd(D) = \max_{\pi \in \Pi_D} |\pi|$$

Finding *wcd*

In this section we describe two methods to calculate the *wcd* of a model, both based on the following observations.

Theorem 1 *If π is non-distinctive, any prefix of π is non-distinctive.*

Proof: According to Definition 1, the fact that π is non-distinctive means that $\exists \pi', \pi'', G', G''$ s.t. $\pi' \in \Pi^*(G')$ and $\pi'' \in \Pi^*(G'')$ and π is a prefix of both π', π''. This in turn means that any prefix of π is necessarily a prefix of both π' and π'', and therefore non-distinctive. ∎

Corollary 1 *If π is distinctive, any path $\pi' \in \Pi^*(\mathcal{G})$ for which π is a prefix, is distinctive.*

Proof: Assume to the contrary that π is distinctive and is a prefix of a non-distinctive path π'. However, according to Theorem 1, if π' is non-distinctive, any prefix of π', including π is non-distinctive, which serves as a contradiction. ∎

The above observations assure us that given a single initial state, any optimal agent will start its progress in the system by following a (possibly empty) non-distinctive path and end with a distinctive path, leading to its goal. This allows us to establish the following relation between a goal recognition design model $D = \langle P, \mathcal{G} \rangle$ and a plan recognition problem, as defined by Ramirez and Geffner (2009), which is achieved by adding to D the set of observations O. $\mathcal{G}^*_O$ represents the set of goals $G \in \mathcal{G}$ such that some optimal plan for G is compatible with O.

Theorem 2 *For a plan recognition model defined by the tuple $T = \langle D, O \rangle$, in which O represents the full sequence of actions performed by an agent, if $|O| >$ wcd(D) then $0 \leq |\mathcal{G}^*_O| \leq 1$.*

Proof: By definition, $wcd(D)$ represents the maximal prefix a set of more than one goal in $\mathcal{G}$ may share. If the set O is the full sequence of observations it represents the prefix of the path of the agent, therefore, if it is bigger than $wcd(D)$, then $0 \leq |\mathcal{G}^*_O| \leq 1$. ∎

A key issue to notice regarding the calculation of the *wcd* value of a model is that, as opposed to classical planning problems in which we are interested in finding any optimal path to the goal, or goal recognition problems in which we are interested in finding any optimal path that fits a set of observations, in our problem we need to take into account all optimal paths to the relevant goals.

A basic way to discover all paths to all goals within a model is to perform an exhaustive exploration of the state space using, for example, a BFS tree search. The BFS tree search starts at the initial state and explores, at each level, all states that are reachable from the previous level. The search continues up to the level in which the most distant goal is found. The result is a tree depicting all paths of length up to the length of the longest optimal path, including all optimal paths to all goals in $\mathcal{G}$. In order to reveal the *wcd* value of the model, we need to find the set of goals that share the longest non-distinctive path. We can do this by performing a backward search starting at the most distant leafs, and advancing one level at time, marking for each node the goals on which it appears on an optimal path, and stop once a node that appears on the path to more than one goal is discovered. Although this method is sound, it is highly inefficient, especially in scenarios in which there are many optimal paths to each goal. We next present two methods for finding the *wcd* of a model, which both rely on Corollary 1 to reduce the state space that is explored.

wcd-bfs

The first solution we present is a variation of the BFS tree search presented above — only instead of blindly exploring all paths in the model we trim the search by pruning the nodes that represent distinctive paths. Corollary 1 assures us that further exploring such a node is futile, since any path that has a distinctive prefix, is also distinctive.

The search starts at the initial state and performs a BFS search in which the nodes of the search graph represent sub-paths and the edges represent the actions that are available at each state. A node representing a distinctive path is marked as solved and is not expanded, otherwise it is added to the queue to be later explored. Notice that we choose the nodes to be sub-paths and not states since we need to find *all* paths that lead to a certain state. The search continues as long as there are nodes in the queue. The *wcd* value of the model is the length of the sub-paths that were expanded at the last iteration of the algorithm.

A node π is pruned once $\mathcal{G}^*_\pi \leq 1$, where $\mathcal{G}^*_\pi$ represents the set of goals to which π is a prefix to an optimal plan. We

find the size of $\mathcal{G}^*_\pi$ by solving a planning problem for each of the goals in $\mathcal{G}^*_{\pi'}$, where π' is the direct predecessor of π. Let $C^*_{s_\pi}(G)$ denote the cost of achieving goal G starting at the state achieved by applying π and let $C(\pi_n)$ denote the cost of applying plan π. A goal $G \in \mathcal{G}^*_{\pi'}$ is added to $\mathcal{G}^*_\pi$ if

$$C^*(G) = C^*_{s_\pi}(G) - C(\pi_n),$$

ensuring that a goal G is added to $\mathcal{G}^*_\pi$ only if π is a prefix to an optimal plan to G.

When comparing this method to the basic search presented above, it is clear that pruning prevents many unnecessary states from being expanded, especially in a domain with a large branching factor. However, using planning several times for each node is expensive and impractical for large domains. One way of improving efficiency is by discovering bounds for the *wcd* of a model that can help trimming the search space further. For such a bound we observe that the *wcd* of a *grd* model is bound from above by the cost of the second most expensive goal. We refer to $\mathcal{G} = \{G_1, \ldots, G_n\}$ as the set of goals, ordered according to the optimal cost of achieving them, where G_1 is the cheapest goal and G_n is the most expensive one. We can therefore state the following.

Theorem 3

$$\text{wcd}(T) \leq C^*(G_{n-1})$$

Proof: For any pair of goals, the longest possible non-distinctive prefix is the optimal path to the less expensive goal. If we choose the two most expensive goals in the model, *i.e.*, G_n and G_{n-1}, we get the two goals for which the possible length of the non distinctive prefix is the maximal. ∎

The above theorem proves an upper bound on the size of the search space $|S|$ to be:

$$|S| \leq b^{c^*(G_{n-1})}$$

where b is the maximal branching factor. This bound can be used to prune any paths with a length that exceeds it and therefore reduces the computational cost. However, the fact that a separate search is performed at each node leaves this method impractical for large domains, as our empirical evaluation shows.

latest-split

The *latest-split* method, which we present in this section, finds the *wcd* of a model by performing a single search for each pair of goals. This is done by compiling the original *grd* problem into a classical planning problem and solving the resulting model with a classical planner.

Given a *grd* problem with a set $\mathcal{G}$ of goals, the *latest-split* method seeks the maximal non-distinctive path a set of agents, each aiming at a separate goal, may share. We start by presenting the application of this idea to a *grd* model where $|\mathcal{G}| = 2$. Next, we extend the discussion to include problems where $|\mathcal{G}| > 2$, and show that the method for $|\mathcal{G}| = 2$ can be used to solve a problem with any number of goals.

We compile the *grd* problem T (with $|\mathcal{G}| = 2$) into a planning problem T', in which we have 2 agents, starting at the same initial state, acting within the same model but each aiming at a different goal. We then solve a planning problem where each agent attempts to achieve its respective goal, but the agents can get a discount for "working together." Following the STRIPS notation in which an action is defined as the set of $\langle pre, add, del \rangle$, we define the *latest-split* compilation as follows.

Definition 3 *For a goal recognition design problem $D = \langle P, \mathcal{G} \rangle$ and cost function C, where $P = \langle F, I, A \rangle$ and $\mathcal{G} = \{G_0, G_1\}$ we create a planning problem $P' = \langle F', I', A', G' \rangle$ and cost function C' that are defined as follows:*

- $F' = \{f_0, f_1 | f_i \in F\} \cup \{\textit{split}\} \cup \{\textit{done}_0\}$
- $I' = \{f_0, f_1 | f_i \in I\}$
- $A' = \{A_{0,1}, A_0, A_1\} \cup \{\textit{DoSplit}\} \cup \{\textit{Done}_0\}$ *where*
  - $A_{0,1}$=$\langle \{f_0, f_1 | f \in pre(a)\} \cup \{\neg \textit{split}\}, \{f_0, f_1 | f \in add(a)\}, \{f_0, f_1 | f \in del(a)\} \rangle | a \in A$
  - A_0=$\langle \{f_0 | f \in pre(a)\} \cup \{\textit{split}\} \cup \{\neg \textit{done}_0\}, \{f_0, | f \in add(a)\}, \{f_0 | f \in del(a)\} \rangle | a \in A$
  - A_1=$\langle \{f_1 | f \in pre(a)\} \cup \{\textit{split}\} \cup \{\textit{done}_0\}, \{f_1, | f \in add(a)\}, \{f_1 | f \in del(a)\} \rangle | a \in A$
  - $\{\textit{DoSplit}\} = \langle \emptyset, \textit{split}, \emptyset \rangle$
  - $\{\textit{Done}_0\} = \langle \textit{split}, \textit{done}_0, \emptyset \rangle$
- $G' = \langle f_0 | f \in G_0 \rangle \cup \langle f_1 | f \in G_1 \rangle$
- C' *s.t.* $C'(a_0) = C'(a_1) = C(a)$, $C'(a_{0,1}) = 2C(a) - \epsilon$

f_i is a copy of F for agent i, *split* is a fluent representing the no-cost action *DoSplit* has occurred, and $done_0$ is a fluent indicating the no-cost $Done_0$ has occurred. The initial state is common to both agents and does not include the $split$ and $done_0$ fluents. Until a *DoSplit* action is performed, the only actions that can be applied are the actions in $A_{0,1}$, which represent the actions the agents perform together. The *DoSplit* action adds $split$ to the current state thus allowing the actions in A_0 to be applied. After agent 0 accomplishes its goal, $Done_0$ is performed, allowing the application of actions in A_1 until G_1 is achieved.

The compiled problem P' is solved using standard classical planning tools that produce an optimal plan $\pi^*_{P'}(G')$ in which each agent i achieves goal G_i. The definition of the model imposes $\pi^*_{P'}(G')$ to start with actions in $A_{0,1}$, after which *DoSplit* occurs, and end with actions in A_0 and A_1, which are the actions that each agent performs separately. This structure is supported by Corollary 1, that assures us that as long as both agents act optimally, after the agents reach a point in which the $DoSplit$ is an optimal choice, they will follow distinctive paths. We enforce agent 1 to wait until agent 0 reaches its goal before starting to act in order to make the search for a solution to P' more efficient by removing symmetries between different interleavings of agent plans after $DoSplit$ occurs.

The *wcd* value of the model is the length of the action sequence until the *DoSplit* action occurs. In order for this value to be correct, the cost function C' for the *latest-split* transformation needs to comply with two objectives. On the one hand, we want to make sure both agents act optimally, namely that the projection of $\pi^*(P')$ on P for

each agent i represents an optimal path in T to G_i. On the other hand, we want to make sure the agents act together as long as possible, ensuring that the prefix of $\pi^*(P')$ until $DoSplit$ occurs is the longest. The way we accomplish both objectives is by making sure that acting together is cheaper than acting separately, but the difference between costs is not enough to cause an agent to diverge from an optimal path in the original problem.

According to the above requirements, we define the cost function C' as one that provides a discount for acting together. C' therefore assigns to $a_{0,1}$ the cost of applying a_i separately for each agent, minus the discount denoted by ϵ. We ensure that agents act optimally in T' in spite of this discount by establishing an upper bound on ϵ.

Theorem 4 *Given a goal recognition design model D and a transformed model T', both agents act optimally in T' if*

$$\epsilon < \frac{1}{c^*(G_{n-1})}$$

Proof: Let π^i_P denote the projection of $\pi^*_{P'}(G')$ on P for agent i. We require that both agents choose a path that is optimal in P, *i.e.*, $\pi^i_P \in \Pi^*(G_i)$ for all i. To ensure this, we require that the difference between the cost of achieving G' in T' and achieving G_0 and G_1 in T is smaller than the cost of the minimal diversion from the optimal paths in T. Assuming an action is associated with a positive cost and that a diversion from an optimal path in T will cost at least 1 we require the following:

$$C'(\pi^*_{P'}(G')) - \sum_i (\pi^*_P(G_i)) < 1$$

Since the difference between the costs of achieving the goals in P' and P is due only to the reduction in costs for the non-distinctive prefix in $\pi^*_{P'}(G')$, whose maximal length is equal to $wcd(T)$. We therefore need to ensure that:

$$\epsilon \cdot wcd(T) < 1$$

and thus when

$$\epsilon < \frac{1}{wcd(T)}$$

the agents will act optimally.

Theorem 3 proves an upper bound on $wcd(T)$. Therefore when

$$\epsilon < \frac{1}{c^*(G_{n-1})}$$

the agents will act optimally. ∎

In the airport example from Figure 1, the *wcd* value is 4 and $c^*(G_{n-1}) = 6$. If $\epsilon < \frac{1}{6} < \frac{1}{4}$, agents will act optimally in T', and therefore reveal the real wcd value of the model.

Multiple goals The *latest-split* compilation finds the *wcd* between a pair of goals. This compilation can be applied to $n > 2$ agents acting within the model. However, in order to integrate n separate searches into a single search, we need to create a copy of the state variables in F for each agent, and of the actions in A for any set of agents that may act separately or together. This may result in a compiled domain with size exponential in n and a planning problem with a branching factor bound by $|A|(2^n - 1)$.

For a more efficient solution we observe that $wcd(T)$ represents the maximal non-distinctive path between all possible sets of goals of size greater than 1. We use $wcd_{\langle G', G'' \rangle}$ to represent the maximal non-distinctive path between two goals G' and G'', that is the maximal non-distinctive path between all pairs π', π'' s.t. $\pi' \in \Pi^*(G')$ and $\pi'' \in \Pi^*(G'')$. We use this notation to state the following :

Lemma 1 *Given a goal recognition design problem $D = \langle P, \mathcal{G} \rangle$,*

$$\text{wcd}(D) = \max_{G', G'' \in \mathcal{G}} (\text{wcd}_{\langle G', G'' \rangle})$$

Proof: According to the definition of a non-distinctive path, if we find the maximal non-distinctive path of a model denoted by π_{wcd}, we are guaranteed to find a set of goals $\mathcal{G}' > 2$ for which π_{wcd} is a prefix of an optimal plan. For any pair of goals $G', G'' \in \mathcal{G}'$, $wcd_{\langle G', G'' \rangle} = wcd(T)$ and since π_{wcd} is maximal, there is no pair of goals that share a longer prefix. ∎

Using Lemma 1, we propose to find the *wcd* of a *grd* problem with $n > 2$, by performing a version of *latest-split* for all goal pairs and assign *wcd* to be the maximal *wcd* of all pairs. This method involves solving $O(n^2)$ planning problems, each with a branching factor of $2|A|$.

Reducing *wcd*

Having defined the *wcd* value as a measure to assess a model and describing ways to calculate it, we turn to our second objective: given a *grd* problem $D = \langle P, \mathcal{G} \rangle$, how to modify D in order to minimize the *wcd* value, while respecting specified constraints. First, we need to select a method to modify the model. A key issue to notice about the selection of the modification method is that in order to have an effect on the *wcd* of the model, any method we choose needs to modify $\bigcup_{G \in \mathcal{G}} \Pi^*(G)$, which is the set of optimal paths to the goals in the model. More specifically, the *wcd* may change only if we remove paths from the set of optimal paths. One way of accomplishing this could be to disallow specific paths in the model. However, our chosen approach consists of the removal of actions from the model as a tool for changing it. Notice that when we disallow a grounded action a from the model we are in fact disallowing the set of paths that include a. Although the proposed technique can only be applied to situations in which the behaviour of the agent can be controlled, we believe removing actions has an appeal in many real-world situations. In our airport example, putting barriers is an easily applicable and common approach for directing the flow of passengers by disallowing specific move actions.

Let D be a *grd* model, $A_\neg$ be the set of actions that are removed from the original model and $D_{A \setminus A_\neg}$ be a *grd* model different from the original model D in that its actions are $A \setminus A_\neg$. Our objective is to minimize the *wcd* of a model without increasing the cost of achieving any of the goals. Given the minimal *wcd* possible, we want to achieve the reduction with the smallest set $A_\neg$. Our objective is therefore expressed by the following bi-objective optimization problem with a

strict ordering between the primary objective to minimize the *wcd* and the secondary objective to minimize the size of the action set.

$$\begin{aligned} \underset{A_\neg}{\text{minimize}} \quad & (wcd(D_{A\setminus A_\neg}), |A_\neg|) \\ \text{subject to} \quad & \forall G \in \mathcal{G}, C^*_D(G) = C^*_{D_{A\setminus A_\neg}}(G) \end{aligned}$$

where $C^*_D(G)$ and $C^*_{D_{A\setminus A_\neg}}(G)$ represent the optimal costs of achieving goal G in D and $D_{A\setminus A_\neg}$, respectively.

Returning to our example from Figure 2, where both presented solutions minimize the *wcd*, we seek a method that produces the solution presented in Figure 2(b), since it requires the elimination of the smallest set of actions. We next present several methods for reducing the *wcd* value of a model while respecting the above requirements.

Exhaustive Search: *exhaustive-reduce*

The first method we present is an exhaustive search, which is a variation of the classical BFS and whose search space are the sets of grounded actions. For a *grd* problem D, a node in the search tree represents a set $A_\neg \subseteq A$, which in turn represents a transformed model $D_{A\setminus A_\neg}$. For each node we calculate the *wcd* and the optimal costs of achieving every goal $G \in \mathcal{G}$ in the corresponding model. The root node of the search is the original model with $A_\neg = \emptyset$, for which we calculate the original *wcd* and optimal costs for all goals. The successors of a node are formed by the concatenation of every action a from A to the set $A_\neg$ of the ancestor node. A node is only explored if the optimal costs to all goals are the same as in the original model. The search continues, increasing at each level of the tree the size of $A_\neg$ until reaching a model in which $wcd = 0$ is found or until there are no more nodes to explore. The result of the search is the set $A^*_\neg$ for which the *wcd* value is minimized. In addition to satisfying the main objective of minimizing the *wcd* of the model, the iterative nature of the algorithm fulfills our secondary requirement by guaranteeing the size of $A_\neg$ is the minimal needed to achieve the reduction. However, in the worst case, the *exhaustive-reduce* method examines all sets of action combinations. Next we seek improvement through pruning, exploiting the characteristics of the *grd* problem.

Pruned Search: *pruned-reduce*

The solution to finding the *wcd* of a *grd* model includes a pair of paths to distinctive goals that share the maximal non-distinctive prefix, which we denote by $\pi^*_{wcd}(D_{A\setminus A_\neg})$. We rely on the fact that when an action is removed from the model, paths that include it are removed as well, to state the following.

Theorem 5 *Given a model D and a transformed model $D_{A\setminus A_\neg}$, if $\forall a \in A_\neg, a \notin \pi^*_{\text{wcd}}(D_{A\setminus A_\neg})$, then* $\text{wcd}(T) = \text{wcd}(T')$.

Proof: The search for the *wcd* value of a model is performed within the set of optimal paths $\bigcup_{G\in\mathcal{G}}\Pi^*(G)$. Since the removal of actions from the model does not create new optimal paths, but only eliminates them, the removal of actions is guaranteed not to add new paths to $\bigcup_{G\in\mathcal{G}}\Pi^*(G)$. Therefore, if the pair of paths in $\pi^*_{wcd}(D_{A\setminus A_\neg})$ are a part of the transformed model $D_{A\setminus a}$ the *wcd* remains unchanged and $wcd(T) = wcd(T')$. ∎

Algorithm 1 *pruned-reduce*

```
wcd= 0 (init)
A*¬ = ∅ (init)
create a list Closed = ∅
create a queue Q initialized to ∅
while Q is not empty:
    A¬ ← Q.dequeue()
    solve(D_{A\A¬})
    Closed ← A¬
    if ∀G ∈ G, C*_D(G) = C*_{D_{A\A¬}}(G)
      if wcd(D_{A\A¬}) < wcd
        wcd = wcd(D_{A\A¬})
        A*¬ = A¬
      for a ∈ π*_wcd(D_{A\A¬}) :
          if a ∪ A¬ not in Closed
              enqueue A¬ ∪ {a} onto Q
return A*¬
```

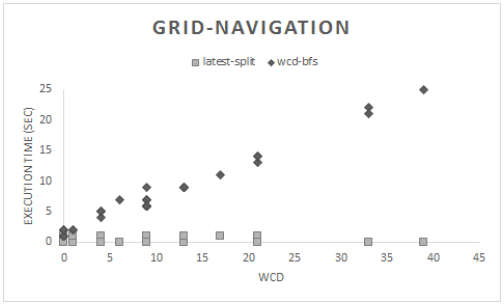
We rely on the above observation to create the *pruned-reduce* algorithm (whose pseudocode is given in Algorithm 1) which is similar to the *wcd-bfs* in that a search node is defined by the set $A_\neg$ it represents. The difference is that instead of creating a successor node for every action in A, a successor node is created only for actions that appear in the optimal solution used to find the *wcd* of the parent node. Since a set of actions may appear in more than one optimal path, we keep a $Closed$ list, of all the action combinations that were previously examined that need not to be solved again.

In order to justify the pruning we perform, we rely on the observation that the removal of an action can only remove existing paths from the model and not introduce new ones. Therefore, the removal of actions is guaranteed not to reduce the optimal cost of achieving any of the goals. At each level of the *pruned-reduce* algorithm we add one action from $\pi^*_{wcd}(D_{A\setminus A_\neg})$ to $A_\neg$, thus removing at least one of the optimal paths to a goal in $\mathcal{G}$. Accordingly, if the optimal cost of achieving at least one goal increases the node may be pruned, since the optimal costs will not decrease for any of its successors.

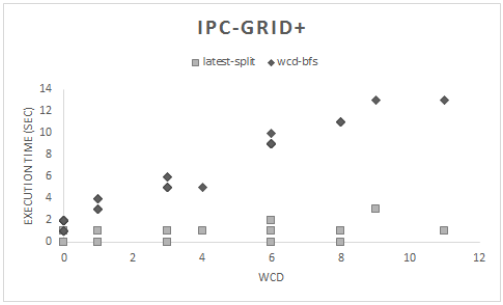
Empirical Evaluation

We now turn to compare the results and performance of the two methods for calculating the *wcd* value, namely *wcd-bfs* and the *latest-split*, and the *exhaustive-reduce* and *pruned-reduce* methods for reducing the *wcd* of a model. We describe first the datasets and the experiment setup before presenting and discussing the experiment results.

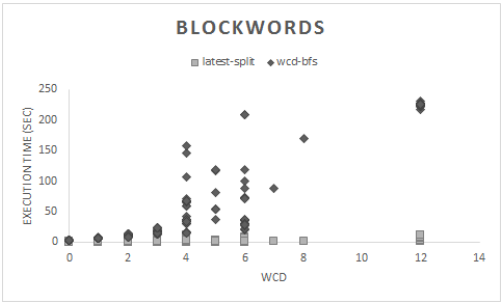
Datasets We perform our evaluations using the benchmarks proposed by Ramirez and Geffner (2009) for plan recognition. This dataset provides a setting where it is not trivial to deduce the goal of an agent. Also, the task of reducing the



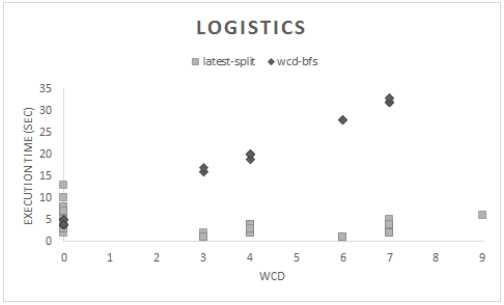
(a) GRID-NAVIGATION



(b) IPC-GRID$^+$



(c) BLOCK-WORDS



(d) LOGISTICS

Figure 3: Calculating *wcd*: individual results

wcd of a model is possible since there may be more than one path to each goal.

The dataset consists of problems from 4 domains, namely GRID-NAVIGATION, IPC-GRID$^+$, BLOCK-WORDS, and LOGISTICS. The problem description contains a domain description, a template for a problem description without the goal, and a set of hypotheses. For each problem we randomly generated hypothesis combinations of varying sizes and created a separate *grd* problem for each combination. All-in-all, we tested 94 GRID-NAVIGATION instances, 163 IPC-GRID$^+$ instances, 233 BLOCK-WORDS instances, and 220 LOGISTICS instances. [1].

Setup For each problem instance, we calculate the *wcd* value by applying both the wcd-bfs and the *latest-split* methods. We measure the *wcd*, execution time, and number of expanded states. The calculation of the reduction of the *wcd* is performed by applying both the *exhaustive-reduce* and *pruned-reduce* (with all suggested pruning) methods with the requirement that the optimal cost of all goals does not increase. Each execution was assigned a time bound of 30 minutes, after which we measured the execution time, number of expanded states, the minimal *wcd* that was achieved and the action set that was removed.

Results Table 1 compares the *wcd-bfs* and the *latest-split* methods. The comparison is broken into domains, comparing execution time, number of expanded states, and percentage of solved problems within the time bound. Each entry in Table 1 represents an average measure over all experiments in the domain. The *latest-split* method outperforms the *wcd-bfs* method in all domains, with up to two orders of magnitude better performance.

Table 1: Calculating *wcd*

	time		expanded		% solved	
	bfs	split	bfs	split	bfs	split
grid navigation	20.3	0.4	928.4	48.1	100	100
ipc-grid+	7.4	1.4	1263.3	2328.7	100	100
logistics	59.6	27.5	51947.8	86596.1	89	92
block-words	45.6	1.9	3816.2	3473.1	99	100

In addition to the aggregated results, we take a look at the individual performance of the two methods. For that purpose, we have compared the execution time of both the *wcd-bfs* and the *latest-split* methods, partitioning it according to domains and *wcd* (for clarity of presentation some outliers were omitted). Figure 3 presents the results for each of the domains. The figure shows that the individual performance follows the aggregated one. In addition, an increase in runtime is observed for both methods with the increase of *wcd*, however this increase is far more prominent with *wcd-bfs* then with *latest-split*.

Table 2 summarizes the set of experiments comparing *exhaustive-reduce* and *pruned-reduce* methods. For each domain we record the percentage of problems that were solved within the time limit, the percentage of problems for which a reduction was found and the average amount of reduction in the *wcd*. The results show that the *pruned-reduce* method solves more problems and is more effective in reducing the *wcd* value.

The results show that in many cases the reduction in *wcd* was possible with the removal of very few actions.

[1]Benchmarks can be found in: http://technion.ac.il/$\sim$sarahn/final-benchmarks-icaps-2014/

Table 2: Reducing *wcd*

	% completed		% reduced		average reduction	
	exhaustive	reduce	exhaustive	reduce	exhaustive	reduce
grid navigation	9	95	18	21	1.64	3.45
ipc-grid+	44	85	28	47	2.07	3.36
logistics	22	86	5	14	2.1	3.46
block-words	11	63	2	9	1	1

For example, the original formulation of the *p5-5-5* problem in the IPC-GRID$^+$ domain with *at-robot place-0-4* and *at-robot place-1-4* as the hypotheses set has *wcd* equal to 4. By disallowing the action *move* from *place-0-2* to *place-1-2*, the *wcd* is reduced to 0 - thus guaranteeing that one step is enough to recognize the goal of any optimal agent.

Related Work

Due to its generic nature, goal recognition has been solved using various approaches and descriptions, including Bayesian networks (Bui 2003; Han and Pereira 2011), graph construction (Hong 2001), and specialized procedures (Lesh and Etzioni 1995), most of which rely on a specification of a plan-library to be supplied. Instead of computing the set of all optimal plans as a plan library, our method relies on a compilation to a classical planning problem to find one optimal plan that encodes the *wcd*.

Despite its similarity to the task of automated planning, which aims to discover plans that lead to a desired goal, it was only recently that the work by Ramirez and Geffner (2009) established the connection between the closely related fields. They present a compilation of plan recognition problems into *classical planning* problems resulting in a STRIPS problem that can be solved by any planner. Several works followed this approach (Agotnes 2010; Pattison and Long 2011; Ramirez and Geffner 2010; 2011) by using various automated planning techniques to analyze and solve plan recognition problems. Although our work uses automated planning models and tools in order to solve the *grd* problem, our problem is a new and different one. Instead of analyzing the behaviour of an agent according to a specific observation sequence, we create a design-time tool that measures and minimizes the maximal number of observations that need to be collected in a fully observable setting before the goal of an agent is recognized.

We are not the first to highlight the influence common prefixes of plans have on the task of goal recognition. Geib (2004) presents a theoretical discussion on the complexity of performing plan recognition on the basis of an analysis of prefixes of plans in a plan library. However, no concrete measure is given that can be used in our model. In the work of Geib (2009), the analysis of plan heads is proposed as a way to increase the efficiency of performing goal recognition given an observation set. Our work relates to a different setting that does not commit to a specific observation sequence and does not rely on a plan library being supplied.

Goal recognition design can be seen as a mechanism design problem, where we want to influence future interactions between the agent and the goal recognition system. Specifically, our approach of reducing the *wcd* by eliminating legal actions can be seen as a social law (Shoham and Tennenholtz 1995), where specific actions can be viewed as being made illegal. This idea was applied in (Agotnes, Van der Hoek, and Wooldridge 2012) where a model is represented by a kripke structure and a social law, defined as the elimination of some of the available transitions, is applied in order to achieve a specified objective. A distance measure between structures is defined to assess the quality of a given law. However, this work offers no method for finding the social laws that fulfill the specified objective. We describe a concrete method for finding the optimal set of actions $A_{\neg}$, whose removal from the model will accomplish the objective of minimizing the *wcd* value.

Conclusion

We presented a new problem we refer to as goal recognition design (*grd*). We introduced the *wcd* value of a *grd* model that represents the maximal number of steps an agent may perform before revealing its final goal. We presented ways of calculating the *wcd* of a model, followed by a presentation of methods for reducing it by disallowing actions from being performed.

The novelty of our approach is in replacing the online analysis of specific observation sequences conventionally performed in goal recognition by a general offline analysis of the goal recognition model. We propose the *wcd* measure that reveals an upper bound on the number of observations that need to be collected in a fully observable setting, before goal recognition can be performed. In addition to providing a crisp definition of the *wcd*, we provide methods for calculating it, one of which is based on a novel compilation to a classical planning problem. The *latest-split* method avoids the need to explore the plan graph and instead finds the *wcd* shared by two goals by solving a single planning problem including two agents, each aiming at a separate goal. This method is used as a basis for calculating the *wcd* for a *grd* problem with multiple goals.

The method we present for reducing the *wcd* of a model is based on the observation that the only actions that should be considered for removal from the model are the ones that appear on optimal plans to the goals. We show that for many of the benchmarks we have tested, the reduction of the *wcd* was possible without increasing the optimal cost of any of the goals.

Acknowledgements

The work was partially supported by the Northeastern-Technion Cooperative Research Program, the DARPA MRC Program, under grant number FA8650-11-C-7192, and Boeing Corporation, under grant number MIT-BA-GTA-1.

References

Agotnes, T.; Van der Hoek, W.; and Wooldridge, M. 2012. Conservative social laws. In *Proceedings of the Twentieth European conference on Artificial Intelligence (ECAI 2012)*, 49–54. IOS Press.

Agotnes, T. 2010. Domain independent goal recognition. In *Stairs 2010: Proceedings of the Fifth Starting AI Researchers Symposium*, volume 222, 238. IOS Press, Incorporated.

Boddy, M. S.; Gohde, J.; Haigh, T.; and Harp, S. A. 2005. Course of action generation for cyber security using classical planning. In *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2005)*, 12–21.

Bui, H. H. 2003. A general model for online probabilistic plan recognition. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*, volume 3, 1309–1315. Citeseer.

Cohen, P. R.; Perrault, C. R.; and Allen, J. F. 1981. Beyond question-answering. Technical report, DTIC Document.

Fikes, R. E., and Nilsson, N. J. 1972. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence* 2(3):189–208.

Geib, C. W., and Steedman, M. 2007. On natural language processing and plan recognition. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 1612–1617.

Geib, C. W. 2004. Assessing the complexity of plan recognition. In *Proceedings of the Nineteenth National Conference of the Amercan Association of Artificial Intelligence (AAAI 2004)*, 507–512.

Geib, C. W. 2009. Delaying commitment in plan recognition using combinatory categorial grammars. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI 2009)*, 1702–1707.

Han, T., and Pereira, L. 2011. Context-dependent incremental intention recognition through bayesian network model construction. In *Proceedings of the Eighth UAI Bayesian Modeling Applications Workshop (UAI-AW 2011)*, volume 818, 50–58. CEUR Workshop Proceedings.

Hong, J. 2001. Goal recognition through goal graph analysis. *Journal of Artificial Intelligence Research(JAIR 2001)* 15:1–30.

Jarvis, P. A.; Lunt, T. F.; and Myers, K. L. 2004. Identifying terrorist activity with ai plan recognition technology. In *Proceedings of the Sixteenth National Conference on Innovative Applications of Artificial Intelligence (IAAI 2004*, 858–863. AAAI Press.

Kaluza, B.; Kaminka, G. A.; and Tambe, M. 2011. Towards detection of suspicious behavior from multiple observations. In *AAAI Workshop on Plan, Activity, and Intent Recognition (PAIR 2011)*.

Kautz, H., and Allen, J. F. 1986. Generalized plan recognition. In *Proceedings of the Fifth National Conference of the Amercan Association of Artificial Intelligence (AAAI 1986)*, volume 86, 32–37.

Kautz, H.; Etzioni, O.; Fox, D.; Weld, D.; and Shastri, L. 2003. Foundations of assisted cognition systems. *University of Washington, Computer Science Department, Technical Report*.

Lesh, N., and Etzioni, O. 1995. A sound and fast goal recognizer. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 1995)*, volume 95, 1704–1710.

Pattison, D., and Long, D. 2011. Accurately determining intermediate and terminal plan states using bayesian goal recognition. *Proceedings of the First Workshop on Goal, Activity and Plan Recognition(GAPRec 2011)* 32.

Ramirez, M., and Geffner, H. 2009. Plan recognition as planning. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI 2009)*.

Ramirez, M., and Geffner, H. 2010. Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI 2010)*.

Ramirez, M., and Geffner, H. 2011. Goal recognition over pomdps: Inferring the intention of a pomdp agent. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence- Volume Three (IJCAI 2011)*, 2009–2014. AAAI Press.

Shoham, Y., and Tennenholtz, M. 1995. On social laws for artificial agent societies: Off-line design. *Artificial Intelligence* 73:231–252.