Goal Recognition Design for Non-Optimal Agents

Sarah Keren and **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

Goal recognition design involves the offline analysis of goal recognition models by formulating measures that assess the ability to perform goal recognition within a model and finding efficient ways to compute and optimize them. In this work we present goal recognition design for non-optimal agents, which extends previous work by accounting for agents that behave non-optimally either intentionally or naïvely. The analysis we present includes a new generalized model for goal recognition design and the worst case distinctiveness (*wcd*) measure. For two special cases of sub-optimal agents we present methods for calculating the wcd, part of which are based on novel compilations to classical planning problems. Our empirical evaluation shows the proposed solutions to be effective in computing and optimizing the *wcd*.

Introduction

Goal recognition design (*grd*) (Keren, Gal, and Karpas 2014) involves the offline analysis of goal recognition models, interchangeably called in the literature plan recognition (Pattison and Long 2011; Kautz and Allen 1986; Cohen, Perrault, and Allen 1981; Lesh and Etzioni 1995; Ramirez and Geffner 2009; Agotnes 2010; Hong 2001), by formulating measures that assess the ability to perform goal recognition within a model and finding efficient ways to compute and optimize them.

Goal recognition design is applicable to any domain for which quickly performing goal recognition is essential and in which the model design can be controlled. Such problems include intrusion detection (Jarvis, Lunt, and Myers 2004; Kaluza, Kaminka, and Tambe 2011; Boddy et al. 2005), assisted cognition (Kautz et al. 2003), and computer games (Kabanza et al. 2010; Albrecht, Zukerman, and Nicholson 1998; Ha et al. 2011). In computer games for example, goal recognition aims at quickly understanding the user's intention in order to enhance her user experience. Goal recognition design offers a tool for designing and modifying these virtual environments, by *e.g.*, preventing certain actions from being performed, in order to guarantee improved real-time goal recognition abilities.

Notice that whereas goal recognition focuses on finding efficient ways to perform the online analysis of incoming observations, goal recognition design is an offline task. Accordingly, the computational efficiency of the *grd* analysis is not a key concern. Instead, the effectiveness of the *grd* analysis is measured according to its ability to asses and minimize the maximal number of observations that need to collected in during the online recognition process.

Previous work in *grd* analysis (Keren, Gal, and Karpas 2014) involves the classification of an observation sequence as *distinctive* if it is a prefix to a plan to only one goal and *non-distinctive* otherwise. Accordingly, the *worst case distinctiveness* (*wcd*) measure is defined as the maximal non-distinctive path.

Keren et al. (2014) offer ways to calculate and minimize the *wcd* of the *grd* model that rely on three simplifying assumptions, namely that the system is assumed to be fully observable, the outcomes of actions are deterministic, and that the agents are assumed to be optimal. These assumptions lead to a compact analysis of the *grd* model and its compilation to classical planning in a straightforward manner.

In this work we relax the *optimality* assumption and offer innovative tools for a *grd* analysis that accounts for non-optimal agents. Non-optimal behavior can be modeled in various ways and can account for settings where agents behave non-optimally, either intentionally or naïvely. We focus our attention on a setting we call *Bounded Non-Optimal*, where an agent is assumed to have a specified budget for diverting from an optimal path. *Bounded Non-Optimal* is suitable for settings where agents are not fully familiar with their environment and may therefore act close to but yet not in a fully optimal manner, as well as for settings where deceptive agents aim at achieving time-sensitive goals, with some flexibility in their schedule.

In addition to exploring the general *Bounded Non-Optimal* case we investigate a special case we call *Bounded Deception* in which an agent behaves non-optimally with the intention of misleading an observer. In this scenario one possible goal is the focus of attention and is referred to as the POI (point of interest) of the system. Agents heading to POI are assumed to have a budget for diverting from an optimal path to their goal, which they use to follow an optimal path to a different goal. In this setting we seek to compute the maximal number of steps an agent aiming at POI can advance on an optimal path to a different goal and still achieve

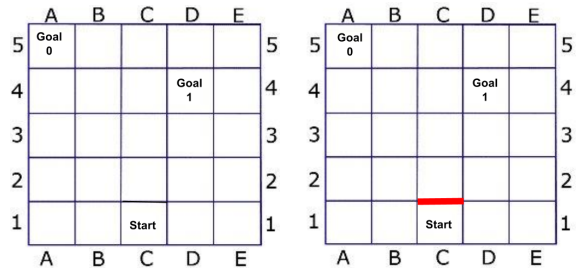


Figure 1: An example of a goal recognition design problem

his goal while respecting the allocated budget.

After calculating the *wcd* of the different settings, it may be desired to minimize it. We implement a procedure called *wcd reduction*, which involves finding the a set of actions whose removal from the model will reduce its *wcd*. This involves searching over subsets of actions, and computing the *wcd* of the model with these actions removed. This procedure was described by Keren, Gal, and Karpas (2014), and we use the same technique here.

To illustrate the objective of calculating and optimizing the *wcd* of a *grd* model, consider the example depicted in Figure 1(a), adopted from (Keren, Gal, and Karpas 2014), which depicts a simplified *grd* problem that 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 0' (domestic flights) and 'Goal 1' (international flights). An agent can move vertically or horizontally from 'Start' to one of the goals. In the optimal setting the $wcd = 3$, referring to a path where an agent advances forward 3 steps before having to turn right to achieve 'Goal 1' or moving on towards 'Goal 0.' In this setting, as depicted in Figure 1(b), a barrier forcing an agent to turn either left or right upon entering room is enough to reduce *wcd* to 0, according to Keren *et al.* (2014). In the bounded non-optimal setting this solution reduces the *wcd* only if the budget of the agents is less than 2.

We use automated planning to model and solve the *grd* problem for non-optimal agents. The advantage of using automated planning tools lies in the availability of established tools and techniques for efficient computation. We show how automated planning can be utilized, despite the sub-optimal nature of agents. Therefore, our main contribution is threefold. First, we show how to generalize the *grd* model to non-optimal agents in a way that would still allow the use of existing techniques for optimizing goal recognition. Secondly, we provide an efficient method for computing *wcd* for non-optimal agents using novel compilation to automated planning. Finally, we provide a thorough empirical analysis to examine the impact of non-optimality on the quality of a goal recognition model.

The rest of the paper is organized as follows. We start by providing background on classical planning. We continue by introducing the formal model representing the *grd* problem for non-optimal agents and the *wcd* value in this setting. The following sections present the methods developed for calculating the *wcd* value of a given *grd* problem. We conclude with an empirical evaluation that shows the effectiveness of the proposed methods, a discussion of related work, and a conclusion.

Background

The basic form of automated planning, referred to as *classical planning*, is a model in which the actions of agents 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 a new state $s' = (s \setminus del(a)) \cup add(a)$. $C : A \rightarrow R^{0+}$ is a cost function that assigns each action a non-negative cost.

The objective of a planning problem 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 $\Sigma_{i=1}^{n}(C(a_i))$. 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. Therefore, plan cost is equivalent to plan length, and the optimal plans are the shortest ones.

grd for Non-Optimal Agents: Model

The goal recognition design (*grd*) problem is defined as

$$D = \langle P_D, \mathcal{G}_D, \Pi_{leg}(\mathcal{G}_D) \rangle$$

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 $g, g \subseteq F$. $\Pi_{leg}(\mathcal{G}_D) = \bigcup_{g \in \mathcal{G}_D} \Pi_{leg}(g)$ is a set of *legal* plans to each of the goals — plans which are allowed under the assumptions we make on the behavior of the agent. $\Pi_{leg}(g)$ may include any path an agent can take to achieve goal g, which can be described either explicitly or symbolically (*e.g.*, the set of all optimal paths that do not make use of action a). Whenever D is clear from the context we use P, $\mathcal{G}$ and $\Pi_{leg}(\mathcal{G})$.

Definition 1, next, defines non-distinctive paths as prefixes of legal plans to goals that belong to more than one goal set. Definition 2 defines *worst case distinctiveness* (*wcd*) to be a value that represent the maximal non-distinctive path in the model.

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

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:*

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

Solving a *grd* problem involves calculating and optimizing the *wcd* of a model. We therefore seek characteristics of the *grd* model that influence the analysis process. One such feature reveals an upper bound on the *wcd* value of a given model. We let $\pi^i_{max} = \max_{\pi \in \Pi_{leg}(g_i)} |\pi|$ be the longest path in $\Pi_{leg}(g_i)$ and $\{g_1, \ldots, g_n\}$ represent the goals in $\mathcal{G}$, ordered according to the increasing lengths of π^i_{max}.

Theorem 1

$$\text{wcd}(T) \leq |\pi^{n-1}_{max}|$$

Proof: For any pair of paths, the length of a non-distinctive path is bound by the length of the shorter path. If we choose g_n and g_{n-1} and consider π^n_{max} and π^{n-1}_{max} we get the pair of paths for which the possible length of the non-distinctive path is maximal and is bound by π^{n-1}_{max}, which is the shorter of the two. ■

We now discuss the effect the removal of actions from the model has on its *wcd* value. We let D and $D^{'}$ represent two *grd* problems. Let A represent the actions of D and $A^{'}$ the actions of $D^{'}$ such that $A^{'} = A \setminus \{a\}$, that is, $A^{'}$ disallows action a. The paths that share the *wcd* of model D are denoted by $\Pi_{wcd}(D)$. Theorem 2 links the reduction of the *wcd* of a model and action removal.

Theorem 2 *Given* grd *models* D *and* $D^{'}$ *s.t.* $A^{'} = A \setminus \{a\}$

$$\begin{cases} \text{wcd}(D^{'}) \leq \text{wcd}(D) & a \in \Pi_{\text{wcd}}(D) \\ \text{wcd}(D^{'}) = \text{wcd}(D) & otherwise \end{cases}$$

Proof: The *wcd* value of a model is defined over the set of legal paths $\Pi_{leg}(\mathcal{G}_D)$. The removal of actions from the model cannot create new paths but only eliminates them. This means that $\Pi_{leg}(\mathcal{G}_D) \subseteq \Pi_{leg}(\mathcal{G}^{'}_D)$ and there cannot be a pair of paths in $\Pi_{leg}(\mathcal{G}^{'}_D)$ that did not exist in $\Pi_{leg}(\mathcal{G}_D)$ and share a longer non-distinctive path. Specifically, if all paths in $\Pi_{wcd}(D)$ do not include action a, $\forall \pi \in \Pi_{wcd}(D), \pi \in \Pi_{wcd}(D^{'})$ and the *wcd* is unchanged. ■

Bounded Non-Optimality

We now introduce a special case of the *grd* model. Among the many ways to represent non-optimal behavior patterns of agents and their corresponding $\Pi_{leg}(\mathcal{G})$, we focus our attention on the *Bounded Non-Optimal* setting, representing a non-optimal behavior where each agent is assigned a budget for diverting from an optimal path.

Definition

We extend the description of a *grd* problem to include a budget specification for each of the goals $B = \langle b_0, ..., b_n \rangle$ where b_i signifies the budget for diverting from an optimal path for an agent aiming at goal g_i. The *Bounded Non-Optimal* formulation of the *grd* problem is therefore

$$D_{bna} = \langle P, \mathcal{G}, \Pi_{leg}(\mathcal{G}), B \rangle$$

A path $\pi \in \Pi_{leg}(g_i)$ if π achieves g_i and $C(\pi) \leq C^*(g_i) + b_i$, where $C^*(g_i)$ is the optimal cost of achieving g_i. Our objective in this setting is to discover the *wcd*, which describes the maximal distance an agent in the system, bounded by the specified budget, can advance without revealing his goal.

One key observation to notice is that for any increase in the budget assigned to any of the goals it is guaranteed that the *wcd* does not decrease. In particular, the *wcd* of the optimal setting, where $\forall i, b_i = 0$ serves as a lower bound for the *wcd* for any assignment of B s.t. $b_i \geq 0$.

Theorem 3 *For any two problems* D_{bna}, $D^{'}_{bna}$ *s.t*

$$D_{bna} = \langle P, \mathcal{G}, \Pi_{leg}(\mathcal{G}) \, , \, B \rangle$$

and

$$D'_{bna} = \langle P, \mathcal{G}, \Pi_{leg}(\mathcal{G}), B' \rangle$$

if $\forall i : b_i \leq b^{'}_i$ *then* $\text{wcd}(D_{bna}) \leq \text{wcd}(D^{'}_{bna})$

Proof: The increase in budget expands the set of legal paths of each goal. In particular, the *wcd* paths, the paths that share the *wcd*, are legal for the extended setting and therefore the *wcd* cannot decrease when the budget increases. ■

Calculating *wcd*

As a baseline for computing the *wcd* in the *Bounded Non-Optimal* setting we use a breadth first search, where each node represents a prefix of a plan. The successors of a node are created by applying each applicable action to the state represented by the parent node. This method, referred to as *wcd-bfs*, prunes a node if it represents a distinctive path. The *wcd* is the length of the longest non-distinctive path that is left in the search.

Following Definition 1, classifying a node π as non-distinctive involves finding a pair of legal paths to two different goals that share π as their prefix. The full observability of the system ensures that each node represents a full sequence of actions starting at the initial state, and therefore a state s_π. Therefore, a path $\pi \in \Pi_{leg}(g_i)$ if $C^*(g_i) + b_i \leq C(\pi) + C^*_{s_\pi}(g_i)$, where $C^*_{s_\pi}(g_i)$ is the cost of a cheapest path from s_π to goal g_i, and the optimal costs are found by solving planning problems.

Compilation to Classical Planning To improve the efficiency of the *wcd* calculation, we exploit the bounded nature of agent suboptimality to compile the problem to a single classical planning problem and solve it using a single search. We first describe the compilation for two agents with a single goal each. Then, we discuss the extension of the technique to $n > 2$ goals.

In a dual goal setting, we create a classical planning problem with two agents, each trying to achieve one goal. Each agent has its own copy of the state propositions so they can be in different states. However, if both agents are in the same state they are encouraged to perform the same action by getting a discount on the action's cost. Additionally, agents can split up, and start performing actions on their own.

The idea of using multiple agents in a single planning problem in order to find the *wcd* was introduced in the *latest-split* compilation (Keren, Gal, and Karpas 2014), which relies on agent optimality. In the *Bounded Non-Optimal* setting, this assumption is no longer appropriate. However, we can still ensure that an optimal solution to the classical planning problem yields the *wcd*, by constraining each agent to take *exactly* $C^*(g_i) + b_i$ actions (including idle actions). An intuitive way to encode the constraints on the number of actions each agent must take is to add a separate counter for the number of actions taken by each agent, and make sure that whenever an agent takes an action, that counter is advanced. We call this compilation the *timed-latest-split*.

Another approach is to include a single global counter. The agents then alternate in taking actions, so that the global counter always counts the number of actions taken by both agents. The restriction on the number of actions each agent is allowed to take can be encoded here, by artificially increasing the cost of one of the goals, so that $C^*(g_0) + b_0 = C^*(g_1) + b_1$, and thus both agents must take the exact same number of actions. We call this compilation *sync-latest-split*. Due to space constraints we will only present the *timed-latest-split* compilation in detail.

timed-latest-split Given a D_{bna} problem where $\mathcal{G} = \{g_0, g_1\}$, the *timed-latest-split* compilation includes two agents: $agent_0$ aiming at g_0 and $agent_1$ aiming at g_1. Agents have two types of actions to choose from. They can either work together by performing 'together-actions,' marked as $A^{0,1}$, which represent the simultaneous execution of action a by both agents. For each action performed together the agents get a discount of ϵ. Alternatively, agents can act alone by performing 'separate-actions' A^i that represent the execution of action a by $agent_i$.

The ordering between the actions in $A^{0,1}$ and A^i is achieved by adding to the model the *DoSplit* no-cost operation, which adds to the current state the *split* predicate. Actions in $A^{0,1}$ are applicable only before *DoSplit* is performed, after which the only applicable actions are the actions in A^i. Since we require agents to act at every stage, we add the *idle* no-op action for each agent. $Idle_i$ is applicable by agent i only after having achieved his goal.

To constrain the path lengths, each agent is assigned a sequence of time steps $\langle time^i_0, ..., time^i_T \rangle$. Each action advances the time step from t to $t+1$ for the acting agents. The goal specification requires that each agent reaches time step $time^i_T$ in addition to achieving his original goal.

Following the STRIPS notation in which an action is defined as the set of $\langle pre, add, del \rangle$, we define the *timed-latest-split* compilation as follows.

Definition 3 *For a* grd *problem*

$$D_{bna} = \langle P, \mathcal{G} = \{g_0, g_1\}, \Pi_{leg}(\mathcal{G}), \{b_0, b_1\} \rangle$$

we create a planning problem $P' = \langle F', I', A', G' \rangle$*, with action costs* C'*, where:*

- $F' = \{f_i, \textit{done}_i \mid f \in F, i \in \{0,1\}\} \cup \{\textit{split}\} \cup \{time^i_t \mid i \in \{0,1\}, t \in \{0...T_i\}\}$
- $I' = \{f_0, f_1 \mid f \in I\} \cup \{time^0_0, time^1_0\}$
- $A' = A^{0,1} \cup A^0 \cup A^1 \cup \{\textit{DoSplit}\} \cup \{Done_i, Idle_i \mid i \in \{0,1\}\}$ *where*
  - $A^{0,1} = \{\langle \{f_0, f_1 \mid f \in pre(a)\} \cup \{\neg \textit{split}\} \cup \{time^i_t\}, \{f_0, f_1 \mid f \in add(a)\} \cup \{time^i_{t+1}\}, \{f_0, f_1 \mid f \in del(a)\} \cup \{time^i_t\} \rangle \mid a \in A\}$
  - $A^i = \{\langle \{f_i \mid f \in pre(a)\} \cup \{split\} \cup \{time^i_t\} \cup \{done_0 \mid i = 1\}, \{f_i \mid f \in add(a)\} \cup \{time^i_{t+1}\}, \{f_i \mid f \in del(a)\} \cup \{time^i_t\} \rangle \mid a \in A\}$
  - $Done_i = \langle \{split\}, \{done_i\}, \emptyset \rangle$
  - $Idle_0 = \langle \{done_0, time^0_t\}, \{time^0_{t+1}\}, \{time^0_t\} \rangle$
  - $Idle_1 = \langle \{done_1, done_0, time^1_t\}, \{time^1_{t+1}\}, \{time^1_t\} \rangle$
  - $DoSplit = \langle \emptyset, \{split\}, \emptyset \rangle$
- $G' = \langle g_0 \cup g_1 \cup time^i_T \rangle$
- $C'(a) = \begin{cases} 2 - \epsilon, & \text{if } a \in A^{0,1} \\ 1, & \text{if } a \in A^i \cup Idle_i \end{cases}$

f_i is a copy of f for agent i. The initial state is common to both agents and does not include $split$. The compiled problem P' is solved using standard classical planning tools that produce an optimal plan $\pi^*_{P'}(G')$. The *wcd* value of the model is the length of the action sequence until the *DoSplit* action occurs.

To encourage the agent to act together, and following (Keren, Gal, and Karpas 2014), the upper bound on ϵ guarantees that agents do not divert from legal paths. According to Theorem 1 the upper bound on ϵ is set according to the bound on the *wcd* of the model s.t. $0 < \epsilon < \frac{1}{|\pi^{n-1}_{max}|}$.

Note that the actions in A^1 include $done_0$ in the precondition specification, enforcing agent 1 to wait until agent 0 reaches its goal before starting to act. This increases efficiency by removing symmetries between different interleaving plans of agents after $DoSplit$ occurs.

Extension to multiple goals The method shown so far finds the *wcd* shared between a pair of goal sets. This compilation can be applied to $n > 2$ goal sets by creating n corresponding agents and integrating them into a single search. However, following the analysis shown for the multiple goal extension presented for the *latest-split* compilation in (Keren, Gal, and Karpas 2014) we will instead perform a separate search for all the pairs and choose the pair with the maximal *wcd*.

Bounded deception as a special case

Having described the general case of bounded non-optimal behavior of agents, we turn our attention to a special case we call *Bounded Deception* in which agents are assumed to be optimal except for agents aiming at g_{poi} that have a specified budget b_{poi} for diverting from an optimal path. The sets of goals excluding g_{poi} are marked as $\bar{\mathcal{G}}$. The definition of the *grd* problem in *Bounded Deception* setting is

$$D_{bd} = \langle P, \bar{\mathcal{G}} \cup g_{poi}, b_{poi} \rangle$$

Our objective is to discover how far an agent aiming at g_{poi}, denoted by $agent_{poi}$, can advance on an optimal path to a

goal in $\bar{\mathcal{G}}$ and still achieve g_{poi} while respecting the specified budget. Otherwise stated, we are trying to discover the *wcd* between $\Pi^*(\bar{\mathcal{G}})$, the set of optimal paths to goal in $\bar{\mathcal{G}}$ and $\Pi_{leg}(g_{poi})$, the plans that achieve g_{poi} with a cost bound by $C^*(g_{poi}) + b_{poi}$.

Compilation for *Bounded Deception* The *Bounded Deception* setting is a special case of the *Bounded Non-Optimal* setting, where $b_i \geq 0$ for g_{poi} and 0 otherwise and can be solved using the techniques presented. Alternatively, we propose to exploit the special structure of the problem to create tailored compilations. The *timed-latest-split-poi* compilation is a variation of the *timed-latest-split*, allowing only timed actions that are performed by $agent_{poi}$, whereas the optimal agent is not constrained by any timing mechanism and does not include references to $time_t^i$.

The variation of the *sync-latest-split* compilation to the *Bounded Deception* setting, which we refer to as *sync-latest-split-poi*, exploits the fact that one of the agents is optimal to remove the timer altogether. Instead, in order to guarantee equal path lengths, agents alternate until the end of execution when it is the turn of the first agent to act. We let g_{opt} represent the goal for which $b_i = 0$ and we denote by $\hat{g_{opt}}$ and $\hat{g_{poi}}$ the goals in our compilation, which have been artificially modified so that $C^*(\hat{g_{opt}}) = \max(C^*(g_{poi}) + b_{poi}, C^*(g_{opt}))$ and $C^*(\hat{g_{poi}}) = C^*(\hat{g_{opt}}) - b_{poi}$. This ensures that both agents have a path length equal to $C^*(g_i)+b_i$.

Empirical Evaluation

Our empirical evaluation has several objectives. Having proven that increased budget may increase the *wcd* value of a model, our first objective is to examine the extent of this effect empirically. Our second objective is to empirically evaluate the two classical planning compilations for the *Bounded Non-Optimal* (BNA) setting, namely *timed-latest-split* ($timed$) and *sync-latest-split* ($sync$), and their specializations to the *Bounded Deception* (BND) setting ($timedPOI$ and $syncPOI$, respectively) on the different settings. The last objective involves examining the ability to reduce the *wcd* by using the technique presented by Keren *et al.* (2014) by eliminating actions from the model in the non-optimal setting.

We examine three settings. In the optimal setting, agents achieve their goals without diverting from optimal paths. In the *Bounded Non-Optimal* setting all agents may have a diversion budget while in the *Bounded Deception* setting all agents are optimal except for a deceptive agent who has a diversion budget from optimal paths.

We first describe the datasets and the experiment setup before presenting and discussing the results.

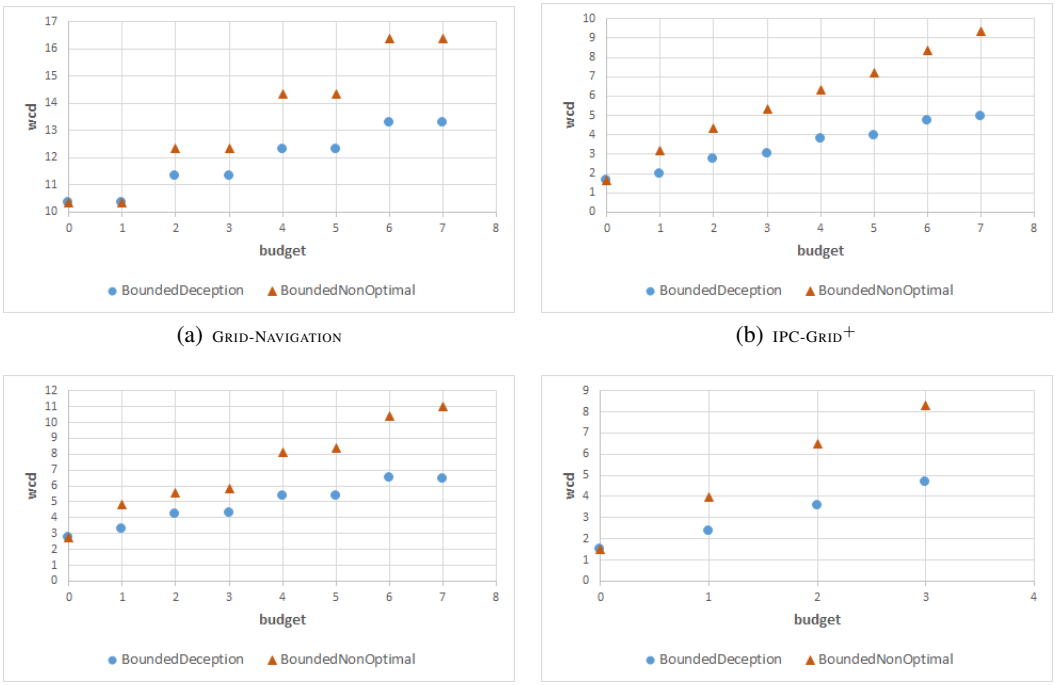
Datasets We use the domains proposed by Ramirez and Geffner (2009) for plan recognition. The dataset consists of problems from 4 domains, namely GRID-NAVIGATION, IPC-GRID$^+$, BLOCK-WORDS and LOGISTICS. For GRID-NAVIGATION and IPC-GRID$^+$ we used all benchmarks proposed by Ramirez and Geffner (2009) and Keren, Gal, and Karpas (2014). For BLOCK-WORDS we randomly selected a subset of the problems in order to keep evaluation time within our time constraints. For LOGISTICS we used smaller instances with goals consisting of single facts instead of conjunctions, as none of the approaches evaluated could handle the original problems with conjunctive goals within the allocated time. Each problem description contains a domain description, a template for a problem description without the goal, and a set of hypotheses. For each problem we generated a separate *grd* problem for each pair of hypotheses. We tested 72 GRID-NAVIGATION instances, 40 IPC-GRID$^+$ instances, 57 BLOCK-WORDS instances, and 40 LOGISTICS instances.

Setup For each problem instance, we calculated the *wcd* value and run-time. For the optimal setting we compared six methods: *latest-split*, *wcd-bfs*, $timedPOI$, $syncPOI$, $timed$, and $sync$. For the *Bounded Deception* setting we compared the former 4 methods and examined problems with the budget of the agents aiming at POI ranging from 1 to 7 for the GRID-NAVIGATION , IPC-GRID$^+$ and BLOCK-WORDS domain and from 1 to 3 for the LOGISTICS (which is the maximal budget the planner could handle). For the *Bounded Non-Optimal* setting we tested the $timed$ and $sync$ methods with the diversion budget of both agents ranging as for the *Bounded Deception* setting. Each execution was assigned a time bound of 30 minutes. For the *wcd* reduction we assigned a time bound of 60 minutes and examined the *Bounded Non-Optimal* setting with an upper bound on the number of actions that could be removed from the model set to 4 and a diversion budget of 4 for each agent in all domains except LOGISTICS, for which we assigned a bound and budget of 2.

Results We First analyze the impact of budget allocation on *wcd*. Figure 2 shows that for all domains, increasing the budget steadily increases the *wcd* value for both the *Bounded Deception* and *Bounded Non-Optimal* settings. *Bounded Non-Optimal* problems consistently yield higher *wcd* values due to the more general nature of the problem that provides agents with higher exploration flexibility and thus an increased worst-case value.

Table 1 summarizes the results for *wcd* run time for the three settings. The comparison is partitioned into settings and into domains. For each setting we compare average run time (in seconds) over commonly solved problems. Whenever some of the problems timed-out, we mention in parenthesis the ratio of solved instances. For the optimal setting *latest-split* outperforms the other method in all domains with up to three orders of magnitude accelaration. For the non-optimal settings, the results for the *wcd-bfs* are not displayed, since for all domains it is outperformed by at least one of the classical planning compilations. In addition, the efficiency achieved by the tailored compilations is more effective for the $sync$ method and almost without effect for $timed$. For both non-optimal settings, the results show the $sync$ compilations to be more efficient in computing the *wcd* for the GRID-NAVIGATION and LOGISTICS domains. The $timed$ compilations are more efficient for the IPC-GRID$^+$ and BLOCKSWORLD domains. The reasons for these performance differences is an open question we intend to investigate in future work.

Table 2 summarizes the results for the *wcd* reduction for the *Bounded Non-Optimal* setting, showing the average



(a) GRID-NAVIGATION (b) IPC-GRID$^+$

(c) BLOCKSWORLD (d) LOGISTICS

Figure 2: *wcd* and execution time given various budget allocations

	Optimal						*Bounded Deception*				*Bounded Non-Optimal*	
	wcd-bfs	*latest-split*	timedPOI	syncPOI	timed	sync	timedPOI	syncPOI	timed	sync	timed	sync
GRID-NAVIGATION	16.78	**0.17**	2.28	0.22	2.42	0.53	2.76	**0.41**	2.78	0.91	4.02	**1.22**
IPC-GRID$^+$	5.82	**1.15**	5.95	1.22	5.57	1.85	**18.39**	23.33	18.44	79.18	**29.85**	616.15(0.84)
LOGISTICS	103.19	**0.7**	372.78	1.8	372.76	2.97	787.43(0.71)	**24.67**	796.58	50.7(0.99)	682.45(0.7)	**480.29**(0.98)
BLOCKSWORLD	36.75	**1.09**	200.35	3.46	200.51	24.3	**192.72**(0.98)	870.61	201.64(0.98)	1035.62(0.84)	**377.11**(0.80)	830(0.30)

Table 1: Average running time for *wcd* calculation over solved problems (when not all solved, the ratio of solved problems are in parenthesis)

wcd reduction achieved within the allocated time, the ratio of problems for which *wcd* was reduced within the allocated budget, and the ratio of problems for which the exploration exhausted all combinations. The evaluation shows that for many of the problems the *wcd* could be decreased, with more then 4 reduced steps for the LOGISTICS domain.

	reduced	average reduction	completed
GRID-NAVIGATION	0.42	1.67	0.58
IPC-GRID$^+$	0.61	2.78	0.24
LOGISTICS	0.87	4.51	0.4
BLOCKSWORLD	0.35	1.3	0.23

Table 2: *wcd* redcution for the *Bounded Non-Optimal* setting

Related Work

Goal recognition design was first introduced by Keren et al. (2014). Our work provides two extensions. First, we relax the optimality assumptions and propose a generic *grd* framework for non-optimal agents and secondly, we offer tools to solve the *grd* model in the *Bounded Non-Optimal* setting.

The first to establish the connection between the closely related fields of automated planning and goal recognition were Ramirez and Geffner (2009). 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 on plan recognition followed this approach (Agotnes 2010; Pattison and Long 2011; Ramirez and Geffner 2010; 2011) by using various automated planning techniques to analyze and solve the problems. Our work exploits the bounded nature of the sub-optimality in the *Bounded Non-Optimal* and *Bounded Deception* settings to create novel compilations of goal recognition design problems into classical planning problems.

Conclusion

We presented a model for goal recognition design for non-optimal agents and the *wcd* measure in this extended setting. We focused our attention on two special cases of sub-optimal agents, namely the *Bounded Non-Optimal* and *Bounded Deception* settings where agents have a budget for diverting from optimal paths, for any goal or one specific goal, respectively. For each of the settings we exploited the bounded nature of the sub-optimality of the agents to create novel compilations to classical planning.

Our empirical evaluation shows that the increase in budget does indeed yield a higher *wcd* value for most of the problems explored. The proposed compilations proved to be effective in computing the *wcd* for all the *grd* problems examined, with different methods excelling in different domains. In addition, we showed that for many of the problems, eliminating actions results in a reduced *wcd*.

Our approach was to provide the minimal possible extension of the *grd* model to support non-optimal agents so that we would still be able to use existing techniques for optimizing goal recognition. When accounting for non-optimal behavior in goal recognition design problems, we increase the model relevancy to a wide range of real world settings. In particular we supply the ability to use optimal classical planning tools for solving *grd* problem for non-optimal settings where it is reasonable to assume the diversion from optimal paths is bounded.

In future work we intend to expand the set of domains that are used to evaluate the non-optimal *grd* setting. In addition, we plan to further investigate the compilation methods presented in the paper in an attempt to study the characteristics of *grd* domains and problems that cause specific calculation methods to outperform the others.

Acknowledgements

The work was carried out in and partially supported by the TechnionMicrosoft Electronic Commerce research center. 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. 2010. Domain independent goal recognition. In *Stairs 2010: Proceedings of the Fifth Starting AI Researchers Symposium*, volume 222, 238. IOS Press, Incorporated.

Albrecht, D. W.; Zukerman, I.; and Nicholson, A. E. 1998. Bayesian models for keyhole plan recognition in an adventure game. *User modeling and user-adapted interaction* 8(1-2):5–47.

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.

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.

Ha, E.; Rowe, J. P.; Mott, B. W.; and Lester, J. C. 2011. Goal recognition with markov logic networks for player-adaptive games. In *AIIDE*.

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.

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

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*.

Keren, S.; Gal, A.; and Karpas, E. 2014. Goal recognition design. In *ICAPS Conference Proceedings*.

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.