Goal Recognition Design with Stochastic Agent Action Outcomes

Christabel Wayllace, Ping Hou, William Yeoh, and Tran Cao Son
New Mexico State University
Las Cruces, NM 88003, USA
{cwayllac, phou, wyeoh, tson}@cs.nmsu.edu

Abstract

Goal Recognition Design (GRD) problems involve identifying the best ways to modify the underlying environment that the agents interact in, typically by making a subset of feasible actions infeasible, in such a way that agents are forced to reveal their goals as early as possible. Thus far, existing work assumes that the outcomes of the actions of the agents are deterministic, which might be unrealistic in real-world problems. For example, wheel slippage in robots can cause the outcomes of their movements to be stochastic. In this paper, we generalize the GRD problem to Stochastic GRD (S-GRD) problems, which handle stochastic action outcomes. We also generalize the worst-case distinctiveness (wcd) measure, which measures the goodness of a solution, to take stochasticity into account. Finally, we introduce Markov decision process (MDP) based algorithms to compute the wcd and minimize it by making up to k actions infeasible.

1 Introduction

Plan and goal recognition problems aim to identify the actual plan or goal of an agent given its behavior. Within the last decade, researchers have made significant progress through synergistic integrations of techniques ranging from natural language processing [Vilain, 1990; Geib and Steedman, 2007] to classical planning [Ramírez and Geffner, 2009; 2010; 2011]. Plan and goal recognition problems have been used to model a number of applications ranging from software personal assistants and robots that anticipate the needs of the users [McQuiggan et al., 2008; Johnson, 2010; Lee et al., 2012; Min et al., 2014]; and security applications that recognize terrorists plans [Jarvis et al., 2005].

The existing research in this area primarily focuses on developing better and more efficient techniques to recognize the plan or the goal of the user given a sequence of observations of the user’s actions. For example, imagine a simplistic security example shown in Figure 1(a), where an agent (e.g., a potential terrorist) is at E3, it can move in any of the four cardinal directions, and its goal is one of three possible goals G1 (at B1), G2 (at A5), and G3 (at C5). Additionally, assume that it will move along a shortest path to its goal. Then, if it moves left to E2, we can deduce that its goal is G1. Similarly, if it moves right to E4, then its goal is either G2 or G3. However, if it is moves up to D3, we cannot make any informed deductions. In fact, if the agent moves along any one of its shortest paths to goal G3, throughout its entire path, which is of length 4, we cannot deduce whether its goal is either G2 or G3! This example illustrates one of the challenges with this approach, that is, there are often a large number of ambiguous observations that can be a result of a large number of goals. As such, it is difficult to uniquely determine the goal of the agent until a long sequence of actions is observed.

Therefore, Keren et al. [2014] very recently proposed an orthogonal approach to modify the underlying environment of the agent, in such a way that the agent is forced to reveal its goal as early as possible. They call this problem the Goal Recognition Design (GRD) problem. For example, if we block the actions (E3, up), (C4, right), (C5, up) in our example problem, where we use tuples (s, a) to denote that action a is blocked from cell s, then the agent can make at most 2 actions (i.e., right to E4 then up to D4) before its goal is conclusively revealed. Figure 1(b) shows the blocked actions. This problem finds itself relevant in many of the same applications of goal recognition because, typically, the underlying environment can be easily modified.

Keren et al. [2014] introduced the notion of worst-case distinctiveness (wcd), as a goodness measure that assesses the ease of performing goal recognition within an environment. The wcd of a problem is the longest sequence of actions an agent can take without revealing its goal. The objective in a GRD problem is then to find a subset of feasible actions to make infeasible such that the resulting
wcd is minimized. In this problem, they make three assumptions: (i) the agents in the system will act optimally (i.e., agents will move along a shortest path to its goal); (ii) the outcomes of the actions of agents are deterministic; and (iii) the environment is fully observable (e.g., the outcomes of the actions can be observed). Since then, they have relaxed some of the assumptions [Keren et al., 2015; 2016a; 2016b]. Additionally, researchers have also addressed the original GRD problem using ASP [Son et al., 2016].

In this paper, we make two extensions to the GRD problem. The first extension limits the maximum number of feasible actions to make infeasible to $k$, which is a user-defined parameter. This extension is identical to the original problem if $k = \infty$. The second extension relaxes the second assumption, that is, we assume that the actions of agents can be stochastic. Our approach lies in the use of Markov decision processes (MDPs) [Mausam and Kolobov, 2012] to formulate the planning problem of the agents within the GRD problem. Our choice is motivated by the fact that MDPs are often de facto models for representing planning problems with uncertainties. Finally, we also introduce new optimization methods that can be used to speed up the search.

2 Background

2.1 Classical Planning

A classical planning model [Geffner and Bonet, 2013], can be represented as a tuple $(S, s_0, A, f, C, G)$, where $S$ is a finite and discrete state space; $s_0$ is the start state of the agent; $A$ is the set of actions, and $A(s)$ is the set of actions applicable in each state $s \in S$; $f : S \times A \rightarrow S$ is a deterministic state transition function, where $s' = f(s, a)$ is the successor state after applying action $a \in A(s)$ in state $s$; $C : A \rightarrow \mathbb{R}^+$ defines the cost for each action; and $G$ is a set of goal states. A plan $\pi = \langle a_1, \ldots, a_n \rangle$ is a sequence of applicable actions that brings an agent from the starting state $s_0$ to a goal state $g \in G$. The cost of a plan $C(\pi) = \sum_i C(a_i)$ is the sum of the cost of each individual action in the plan. The goal is typically to find a cost-minimal plan $\pi^* = \text{argmin}_\pi C(\pi)$.

2.2 Goal Recognition Design (GRD)

A Goal Recognition Design (GRD) problem [Keren et al., 2014] is represented as a tuple $P = \langle D, G \rangle$, where $D = \langle S, s_0, A, f, C \rangle$ captures the domain information and $G$ is a set of possible goal states of the agent. The elements in the tuple $D$ are as they are described in classical planning except that all actions have the same cost of 1. The worst case distinctiveness (wcd) of problem $P$ is the length of a longest sequence of actions $\pi = \langle a_1, \ldots, a_k \rangle$ that is the prefix in cost-minimal plans $\pi^*_g$ and $\pi^*_g$ to distinct goals $g_1, g_2 \in G$.

Using our example problem of Figure 1(a), a longest sequence of actions that can lead to two distinct goals is $\langle (E3, up), (D3, up), (C3, right), (C4, right) \rangle$, where we use tuples $(s, a)$ to denote that action $a$ is taken from cell $s$. This sequence of actions can lead to either goals G2 or G3 and is of length 4. Thus, the wcd of the problem is 4.

The objective in GRD is to find a subset of actions $A^* \subset A$ such that if they are removed from the set of actions $A$, then the wcd of the resulting problem is minimized. This optimization problem is subject to the requirement that the cost of cost-minimal plans to achieve each goal $g \in G$ is the same before and after removing the subset of actions. More specifically, the objective is to find:

$$A^* = \text{argmin}_{A} \text{wcd}(\hat{P})$$

subject to $C(\pi^*_g) = C(\hat{\pi}^*_g) \quad \forall g \in G$

where $\hat{P} = \langle \hat{D}, \hat{G} \rangle$ is the problem with the resulting domain $\hat{D} = \langle S, s_0, A \setminus \hat{A}, f, C \rangle$ after removing actions $A$, $\pi^*_g$ is a cost-minimal plan to achieve goal $g$ in the original problem $P$, and $\hat{\pi}^*_g$ is a cost-minimal plan to achieve goal $g$ in problem $\hat{P}$. For example, blocking the actions $(E3, up), (C4, right), (C5, up)$ in our example problem, where we use tuples $(s, a)$ to denote that action $a$ is blocked from cell $s$, will reduce the wcd of the problem to 2. Figure 1(b) shows the actions blocked. Then, the longest sequence of actions is $\langle (E3, right), (E4, up) \rangle$, which can lead to goals G2 or G3. Thus, the resulting wcd is 2.

2.3 Markov Decision Process (MDP)

A Stochastic Shortest Path Markov Decision Process (SSP-MDP) [Mausam and Kolobov, 2012] is represented as a tuple $(S, s_0, A, T, C, G)$. It consists of a set of states $S$: a start state $s_0 \in S$; a set of actions $A$: a transition function $T : S \times A \times S \rightarrow [0, 1]$ that gives the probability $T(s, a, s')$ of transitioning from state $s$ to $s'$ when action $a$ is executed; a cost function $C : S \times A \times S \rightarrow \mathbb{R}^+$ that gives the cost $C(s, a, s')$ of executing action $a$ in state $s$ and arriving in state $s'$; and a set of goal states $G \subseteq S$. The goal states are terminal, that is, $T(s, a, g) = 1$ and $C(s, a, g) = 0$ for all goal states $g \in G$ and actions $a \in A$.

An SSP-MDP must also satisfy the following two conditions: (1) There must exist a proper policy, which is a mapping from states to actions with which an agent can reach a goal state from any state with probability 1. (2) Every improper policy must incur an accumulated cost of $\infty$ from all states from which it cannot reach the goal with probability 1. In this paper, we will focus on SSP-MDPs and will thus use the term MDPs to refer to SSP-MDPs.

A “solution” to an MDP is a policy $\pi$, which maps states to actions. Solving an MDP is to find an optimal policy, that is, a policy with the smallest expected cost. Value Iteration (VI) [Bellman, 1957] is one of the fundamental algorithms to find an optimal policy. It uses a value function $V$ to represent expected costs. The expected cost of an optimal policy $\pi^*$ for the starting state $s_0 \in S$ is the expected cost $V(s_0)$, and the expected cost $V(s)$ for all states $s \in S$ is calculated using the Bellman equation [Bellman, 1957]:

$$V(s) = \min_{a \in A} \sum_{s' \in S} T(s, a, s') [C(s, a, s') + V(s')]$$

The action chosen by the policy for each state $s$ is then the one that minimizes $V(s)$.

3 Stochastic GRD Problem

We now describe the Stochastic GRD (S-GRD) problem, which is an extension of the original GRD problem along two
dimensions. First, it limits the maximum number of actions to make infeasible to a user-defined parameter \( k \). Second, it handles uncertainty in the actions of agents. In other words, an agent executing an action can now transition to multiple possible successor states with some probability. This extension is important because it can better model some real-world applications. For example, assume that the agent in our example in Figure 1 is a robot. Then, due to slippage, the robot has a small probability of staying where it is when it attempts to move. This characteristic, which cannot be captured by the original GRD problem, is captured in the S-GRD problem.

An S-GRD problem is also represented as a tuple \( P = (D, G) \) similar to GRD problems, except that the domain information \( D = (S, s_0, A, T, C) \) is now represented similar to an MDP instead of a classical planning problem. Therefore, the elements in the tuple \( D \) are as they are described in MDPs. Like in the original GRD problem, we also assume here that all actions have the same cost of 1 though our model can be generalized to handle non-uniform costs.

Definition 1 (policy prefix) A policy prefix is a mapping \( \pi : S_\pi \rightarrow A \), where \( s_0 \in S_\pi \subseteq S \) and, for each \( s \in S_\pi \setminus \{s_0\} \), there exists a state sequence \( (s_0, s_1, \ldots, s_n = s) \), where \( T(s_i, \pi(s_i), s_{i+1}) \geq 0 \) and \( s_i \in S_\pi \).

In other words, a policy prefix is a policy defined only on some states that are reachable from the start state \( s_0 \). Figures 3(a) and 3(b) show examples of policy prefixes for an optimal policy to reach goal G2.

Definition 2 (boundary states) Given a policy prefix \( \pi : S_\pi \rightarrow A \), the set of boundary states \( B_\pi \) is defined as:

\[
B_\pi = \{ s \mid \exists T(s', \pi(s'), s) > 0 \land s' \in S_\pi \land s \notin S_\pi \}
\]

In other words, the set of boundary states is composed of all states \( s \) that are undefined in the policy prefix and whose predecessor state \( s' \) is defined in the policy prefix.

Definition 3 (non-distinctive policy prefix) Given a problem \( P = (D, G) \), a policy prefix \( \pi \) is a non-distinctive policy prefix in \( P \) if \( \exists g', g'' \in G \) such that \( g' \neq g'' \) and \( \pi(s) = \pi_{g'}(s) = \pi_{g''}(s) \) for all states \( s \in S_\pi \), where \( \pi_{g'} \) and \( \pi_{g''} \) are optimal policies for the problems \( (D, g') \) and \( (D, g'') \), respectively, and the goals \( g' \) and \( g'' \) are terminal states. Otherwise, \( \pi \) is distinctive.

Intuitively, as long as an agent executes a non-distinctive policy prefix \( \pi \), it does not reveal its goal. However, the agent might eventually transition to a boundary state \( b \in B_\pi \). If it transitions to goal boundary state \( b \in G \) and stops executing any actions, then it has revealed its goal to be state \( b \). If it transitions to a non-goal boundary state \( b \notin G \) and executes a particular action in that state, it might have revealed that its goal is not one of the two goals that define policy prefix \( \pi \) in Definition 3. For instance, consider distinct goals G2 and G3 in the example problem given in Figure 1. If the actions can either succeed with probability \( p \) or have no effect with probability \( 1 - p \), then Figure 2(a) shows the set of actions (denoted by arrows) that form all possible non-distinctive policy prefixes to those two goals and Figure 2(b) shows an example of one non-distinctive policy prefix and the corresponding set of boundary states (cells shaded in dark grey). In this case, there is only one boundary state for this policy prefix.

We now need a metric to assess the largest number of actions an agent can take or, equivalently, the largest cost an agent can incur before revealing its goal. Since transitions are stochastic, the number of actions or, equivalently, the cost incurred can be measured either in in the worst-case maximum or in the expectation. The worst-case maximum may be infinite (e.g., when the agent can transition back to its previous state and, thus, potentially get stuck in a loop), which prohibits meaningful comparisons. Therefore, in this paper, we choose to measure in the expectation and use the worst-case distinctiveness (wcd) as this measure.

Before defining the wcd, let \( V_\pi(s_0) \) denote the expected cost of reaching a boundary state \( b \in B_\pi \) from the start state with policy prefix \( \pi \). It is recursively computed using:

\[
V_\pi(s) = \sum_{s' \in S} T(s, \pi(s), s') [C(s, \pi(s), s') + V_\pi(s')] \quad (3)
\]

Definition 4 (worst-case distinctiveness) The worst-case distinctiveness (wcd) of a problem \( P \) is defined as:

\[
wcd(P) = \max_{\pi \in \Pi} V_\pi(s_0) \quad (4)
\]

where \( \Pi \) is the set of all non-distinctive policy prefixes and \( V_\pi(s_0) \) is recursively computed using Equation 3.

In other words, the wcd of a problem is the largest expected cost to reach a boundary state from the start state over all possible non-distinctive policy prefixes.

The objective in S-GRD problems is identical to the objective in the original GRD problems except that wcd computations are now done according to Definition 4 and costs are measured in the expectation. More formally, the objective is to find a subset of actions \( \hat{A}^* \subset A \):

\[
\hat{A}^* = \arg\min_{\hat{A} \subset A} \text{wcd}(\hat{P})
\]

subject to \( V_{\pi_g^*}(s_0) = V_{\pi_g^*}(s_0) \quad \forall g \in G \) \quad (5)

where \( \hat{P} = (\hat{D}, G) \) is the problem with the domain \( \hat{D} = (S, s_0, \hat{A} \setminus \hat{A}, T, C) \) after removing actions \( \hat{A} \), \( \pi_g^* \) is an optimal policy to achieve goal \( g \) in the original problem \( P \), and \( \pi_g^* \) is an optimal policy to achieve goal \( g \) in problem \( \hat{P} \).

\[1\] The evaluation is done on an MDP where all the boundary states are terminal goal states.
Algorithm 1: FIND-WCD(P = (D, G))

1. \( wcd \leftarrow 0 \)
2. for \( g \in G \) do
   3. \( \text{SOLVE-MDP}(D, g) \)
   4. \( \Pi_g^* \leftarrow \{ \pi_g^* \mid \pi_g^* \text{ is an optimal policy to reach } g \} \)
5. for \( g, g' \in G \) do
   6. \( \text{A}_{g,g'}^{\text{cand}} \leftarrow \emptyset \)
   7. for \( s \in S, \pi_g^* \in \Pi_g^* \), and \( \pi_{g'}^* \in \Pi_{g'}^* \) do
      8. if \( s \) is a reachable state with \( \pi_g^* \) and \( \pi_{g'}^* \) and
         \( \pi_g^*(s) = \pi_{g'}^*(s) \) then
         9. \( \text{A}_{g,g'}^{\text{common}} \leftarrow \text{A}_{g,g'}^{\text{common}} \cup \{ (s, \pi_g^*(s)) \} \)
   10. if \( s \) is a reachable state with \( \pi_{g'}^* \) then
      11. \( \text{A}_{g,g'}^{\text{cand}} \leftarrow \text{A}_{g,g'}^{\text{cand}} \cup \{ (s, \pi_{g'}^*(s)) \} \)
   12. if \( s \) is a reachable state with \( \pi_{g}^* \) then
      13. \( \text{A}_{g,g'}^{\text{common}} \leftarrow \text{A}_{g,g'}^{\text{common}} \cup \{ (s, \pi_{g}^*(s)) \} \)
   14. \( \text{wcd}_{g,g'} \leftarrow 0 \)
   15. \( \Pi_{g,g'}^{\text{common}} \leftarrow \text{CONSTRUCTPOLICIES}(\text{A}_{g,g'}^{\text{cand}}) \)
   16. for \( \pi \in \Pi_{g,g'}^{\text{common}} \) do
      17. \( V_{\pi} \leftarrow \text{CALCEXPECTEDCOST}(\pi, D, B_n) \)
      18. if \( \text{wcd}_{g,g'} < V_{\pi} \) then
         19. \( \text{wcd}_{g,g'} \leftarrow V_{\pi} \)
      20. \( \text{wcd} \leftarrow \text{wcd}_{g,g'} \)
      21. return \( \langle \text{wcd}, \{ \text{wcd}_{g,g'} \}_{g,g' \in G}, \{ \text{A}_{g,g'}^{\text{cand}} \}_{g,g' \in G} \rangle \)

Theorem 1 The Stochastic GRD (S-GRD) problem subsumes the (deterministic) GRD problem when \( k = \infty \).

3.1 Finding the \( wcd \)

We now introduce FIND-WCD, which computes the \( wcd \) of a given problem. Algorithm 1 shows the pseudocode of this algorithm. Essentially, it follows the logic in Definition 4 via the following high-level steps: (1) Find the set of all possible non-distinctive policies \( \Pi \); (2) Calculate the expected cost \( V_{\pi}(s_0) \) of each non-distinctive policy \( \pi \in \Pi \) using the Bellman equation; and (3) Return the largest expected cost, which is also the \( wcd \) of the problem. We now describe these steps in more detail:

- **Step 1:** In order to find the set \( \Pi \), the algorithm first solves \( |G| \) MDPs, where each MDP models the problem that contains only a single terminal goal \( g \in G \), and stores all optimal policies for each MDP in \( \Pi_g^* \) (lines 2-4). For example, if we model our example deterministic problem of Figure 1 with MDPs, then Figure 3 shows two optimal policies to move to goal G2 (Figures 3(a) and 3(b)) and goal G3 (Figures 3(c) and 3(d)).

- **Step 2:** In order to find the set of candidate sets, it first finds the set \( \text{A}_{g,g'}^{\text{cand}} \) for each pair of distinct goals \( g, g' \in G \) and takes the union of all these sets over all goals \( g, g' \in G \) (line 25). The construction of each set \( \text{A}_{g,g'}^{\text{cand}} \) is done in FIND-WCD (lines 10-13). The set contains the state-action pairs that are in the optimal policies to either goals \( g \) or \( g' \) and are reachable with the optimal policies. State-action pairs that are not in the optimal policies or are not reachable need not be considered since removing them will not reduce the \( wcd \). For example, Figures 3(f) and 3(g) show \( \text{A}_{g,g'}^{\text{cand}} \) in our example problem.

After finding all these sets in FIND-WCD, the algorithm takes the union of all these sets over all pairs of distinct goals \( g, g' \in G \) and stores the union in \( \text{A}_{g,g'}^{\text{common}} \) (line 25). This union now contains all candidate state-action pairs that can be removed. For example, Figure 3(h) shows \( \text{A}_{g,g'}^{\text{cand}} \) in our example problem. If \( k = 1 \),

2To be precise, it does not include non-distinctive policies that are sub-policies of another non-distinctive policy since they do not affect the correctness of the \( wcd \) computations.
Figure 3: Example Partial Trace

Algorithm 2: \textsc{Reduce-WCD}(P = (D, G), k)

\begin{algorithm}
\begin{algorithmic}[1]
\State $\langle \text{wcd}, \text{wcd}_{g,g'} \rangle, g, g' \in G. (A^{cand}_{g,g'}) \leftarrow \text{FIND-WCD}(P)$
\State $A^* \leftarrow \emptyset$
\State $A^{cand} \leftarrow \bigcup_{g, g' \in G} A^{cand}_{g,g'}$
\State $Q \leftarrow$ set of all $i \leq k$ state-action combinations from $A^{cand}$
\While{$Q \neq \emptyset$}
\State $A \leftarrow$ an element in $Q$
\State $Q \leftarrow Q \setminus A$
\If{$\exists (s, \hat{a}) : (\hat{s}, \hat{a}) \in A \Land (s, \hat{a}) \in A^{cand}$}
\State $D \rightarrow \langle S, s_0, T, A \setminus A, C \rangle$
\For{$g \in G$}
\State $V^*(s_0) \leftarrow \text{SOLVE-MDP}(D, g)$
\State $\hat{V}^*(s_0) \leftarrow \text{SOLVE-MDP}(\hat{D}, g)$
\If{$V^*(s_0) \neq \hat{V}^*(s_0)$}
\State \text{go to Line 27}
\EndIf
\EndFor
\State $\langle \text{wcd}_{A^*}, \text{wcd} \rangle \leftarrow \text{FIND-WCD}(\langle D, G \rangle)$
\If{$\text{wcd}_{A^*} < \text{wcd}$}
\State $\text{wcd} \leftarrow \text{wcd}_{A^*}$
\EndIf
\State $A^* \leftarrow A$
\EndIf
\EndWhile
\State \Return{$\langle A^*, \text{wcd} \rangle$}
\end{algorithmic}
\end{algorithm}

one can iterate through all state-action pairs in this set and check for the one that minimizes the wcd. However, if $k > 1$, the algorithm constructs a set $Q$, which contains all $\bigcup_{i=1}^{k} (A^{cand}_i)$ combinations of state-action pairs (line 26).

- **Step 3:** We first describe an unoptimized version of this step, which skips line 30, and describe the optimized version later. The algorithm evaluates each set of (up to) $k$ state-action pairs $A$ in the candidate set $Q$ and ensures that the removal of these state-action pairs will not change the expected cost to reach any of the goals $g \in G$ (lines 27-29, 31-36). If it changes the expected cost for any goal, these state-action pairs are not considered and the algorithm iterates to the next set of state-action pairs. If it does not change the expected cost for any goal, then the algorithm computes the wcd if these state-action pairs are removed (line 37). Note that the \text{FIND-WCD} function call here should not include lines 10-13.

- **Step 4:** Finally, it stores the best wcd reduction and the corresponding set of state-action pairs (lines 38-40) and returns it (line 41).

We now describe the optimization on line 30 for Step 3. For each pair of distinct goals $g, g' \in G$, the algorithm computes the wcd of a subproblem that considers only these goals and ignores all other goals (lines 18-19). This wcd value is stored in variable $\text{wcd}_{g,g'}$. Additionally, observe that in order to reduce the wcd of the original problem, then at least one state-action pair in candidate set $A^{cand}$ must be removed for all pairs of distinct goals $g, g' \in G$, where $\text{wcd}_{g,g'}$ is the wcd of the original problem. Otherwise, all the original actions that can lead an agent to either goals are not removed, and the wcd will remain the same. For example, if $k = 3$, then removing the actions $\{(E2, up), (D2, up), (C2, up)\}$ will not reduce the wcd since none of those actions are in $A^{cand}_{G2,G3}$ and $\text{wcd}_{G2,G3}$ equals the wcd of the problem.

Theorem 2 formalizes the optimization on line 30, where the algorithm ignores candidate $k$ state-action pairs that do not include at least one state-action pair in $A^{cand}_{g,g'}$ for all goals $g, g' \in G$ whose $\text{wcd}_{g,g'}$ is greater than the current best wcd. We believe this optimization will also apply to other search-based algorithms that searches the space of actions to reduce the wcd, such as those proposed by [Keren et al., 2014].

**Theorem 2** The wcd of the problem wcd*$ will not be reduced by removing up to $k$ state-action pairs $A$, where $\exists (s, a) \in A : (s, a) \in A^{cand}_{g,g'} \Wand wcd_{g,g'} \geq wcd^*$ $\Wand g \neq g'$. 


Finally, solving S-GRDs is at least NP-hard as one of the subroutines, solving the Stochastic Longest Path (SLP) problem to compute the \textit{wcd}, is NP-hard [Kolobov, 2013].

## 4 Experimental Results

We evaluated our REDUCE-WCD algorithm with and without the optimization on Line 30 (labeled \textit{R-W(o)}) and \textit{R-W(¬o)}, respectively, on the same four deterministic benchmarks domains [Keren et al., 2014], except that we modified them to allow for stochastic actions, where each action can transition to its deterministic successor with probability 0.9 and stay in the same state with probability 0.1. The four domains are: (1) GRID-Navigation, where each instance is defined by the \(x\)- and \(y\)-dimensions and the number of cells; (2) IPC-GRID\(^+\), where each instance is defined by the \(x\)- and \(y\)-dimensions and the number of locks/keys; (3) BLOCK-WORDS, where each instance is defined by the number of blocks and words/goals; and (4) LOGISTICS, where each instance is defined by the number of packets, places, trucks, and airplanes. Table 1 tabulates the results. The experiments were conducted on a 3.1GHz quad-core machine with 6GB of RAM and a timeout of 2 days was set.

As expected, when \(k\) increases, the runtimes increased but the \textit{wcd} was reduced in more cases. For the instances where the \textit{wcd} remained unchanged, we suspect that it is because of one of the following reasons: (1) the \textit{wcd} cannot be reduced further even if \(k = \infty\), or (2) the \textit{wcd} can only be reduced with a significantly larger \(k\). For the second and third GRID-Navigation instances, we have empirically verified that it is the former case as there is only one optimal policy to reach each of the goal and blocking any action will result in a suboptimal policy. This is also the reason why the runtimes for these instances are very small. For the other instances, especially those with significantly larger runtimes, it may be the latter case. However, since S-GRDs are \textit{offline} problems, if a larger \(k\) is necessary, one could run the algorithm for a longer amount of time.

- GRID-NAVIGATION and IPC-GRID\(^+\) are the two domains with a larger number of instances with \textit{wcd} reductions. In the BLOCK-WORDS domain, words (goals) can differ in only one letter. As a result, the goals are very similar to one another and it is difficult to reduce the \textit{wcd} with small values of \(k\). In the LOGISTICS domain, the optimal policies for the different goals are already rather distinct. As a result, it is also difficult to reduce the \textit{wcd}.
- In general, the runtimes of \textit{R-W(o)} are smaller than the runtimes of \textit{R-W(¬o)} due to the optimization savings. These savings are up to 82%. However, the runtimes of the \textit{R-W(o)} can be larger than the runtimes of \textit{R-W(¬o)}, indicating that in those instances, little pruning is done and time is wasted on overhead. The additional overhead results in at most 8% more runtime.
- While the runtime of \textit{R-W} can be large, this is not a major concern since S-GRD problems are offline problems.

### 5 Conclusions and Future Work

A new and novel approach to goal recognition is \textit{Goal Recognition Design} (GRD), where the problem is to modify the underlying environment such that agents are forced to reveal their goals as early as possible. However, the GRD model and existing GRD algorithms assume that the outcomes of agent actions are deterministic. In this paper, we propose the \textit{Stochastic GRD} (S-GRD) model, which subsumes GRD and allows outcomes of agent actions to be stochastic, as well as introduce an MDP-based approach to solve this problem.

For future work, we will consider other metrics aside from \textit{wcd} such as those that measure the mean and variance of the distribution of the expected cost over all possible non-distinctive policy prefixes. We will also consider other possible definitions for non-distinctive policy prefixes such as those that are defined over \textit{all} goals instead of \textit{pairs} of goals.

### Table 1: Experimental Results for Stochastic GRD Problems

<table>
<thead>
<tr>
<th>Domain</th>
<th>Instances</th>
<th>\textit{wcd} Reduction</th>
<th>Runtime (s)</th>
<th>\textit{wcd} Reduction</th>
<th>Runtime (s)</th>
<th>\textit{wcd} Reduction</th>
<th>Runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) (k = 1)</td>
<td>4-12-48</td>
<td>10.0 (\rightarrow) 10.0</td>
<td>223</td>
<td>217</td>
<td>10.0 (\rightarrow) 8.9</td>
<td>3,933</td>
<td>3,909</td>
</tr>
<tr>
<td></td>
<td>8-18-39</td>
<td>18.9 (\rightarrow) 18.9</td>
<td>1</td>
<td>1</td>
<td>18.9 (\rightarrow) 18.9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>10-7-55</td>
<td>43.3 (\rightarrow) 43.3</td>
<td>2</td>
<td>2</td>
<td>43.3 (\rightarrow) 43.3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>9-15-44</td>
<td>4.4 (\rightarrow) 4.4</td>
<td>1</td>
<td>1</td>
<td>4.4 (\rightarrow) 3.3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>(b) (k = 2)</td>
<td>5-5-4</td>
<td>4.44 (\rightarrow) 2.22</td>
<td>179</td>
<td>179</td>
<td>4.44 (\rightarrow) 2.22</td>
<td>294</td>
<td>226</td>
</tr>
<tr>
<td></td>
<td>10-5-4</td>
<td>13.33 (\rightarrow) 11.11</td>
<td>2,097</td>
<td>2,078</td>
<td>13.33 (\rightarrow) 11.11</td>
<td>3,327</td>
<td>3,328</td>
</tr>
<tr>
<td></td>
<td>5-10-4</td>
<td>12.22 (\rightarrow) 8.89</td>
<td>3,864</td>
<td>3,842</td>
<td>12.22 (\rightarrow) 8.89</td>
<td>4,984</td>
<td>4,241</td>
</tr>
<tr>
<td></td>
<td>10-10-4</td>
<td>20.00 (\rightarrow) 18.89</td>
<td>112,205</td>
<td>111,842</td>
<td>20.00 (\rightarrow) 15.56</td>
<td>117,207</td>
<td>114,449</td>
</tr>
<tr>
<td>(c) (k = 3)</td>
<td>5-3</td>
<td>3.33 (\rightarrow) 3.33</td>
<td>4</td>
<td>4</td>
<td>3.33 (\rightarrow) 3.33</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>6-5</td>
<td>4.44 (\rightarrow) 2.22</td>
<td>376</td>
<td>321</td>
<td>4.44 (\rightarrow) 2.22</td>
<td>4,082</td>
<td>718</td>
</tr>
<tr>
<td></td>
<td>7-8</td>
<td>11.11 (\rightarrow) 11.11</td>
<td>13,649</td>
<td>13,619</td>
<td>—</td>
<td>timeout</td>
<td>timeout</td>
</tr>
<tr>
<td></td>
<td>7-11</td>
<td>7.78 (\rightarrow) 7.78</td>
<td>21,268</td>
<td>5,781</td>
<td>7.78 (\rightarrow) 7.78</td>
<td>159,924</td>
<td>—</td>
</tr>
<tr>
<td>(a) (k = 1)</td>
<td>2-2-2-2</td>
<td>6.67 (\rightarrow) 6.67</td>
<td>4</td>
<td>4</td>
<td>6.67 (\rightarrow) 6.67</td>
<td>29</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>2-3-3-3</td>
<td>6.67 (\rightarrow) 6.67</td>
<td>112</td>
<td>111</td>
<td>6.67 (\rightarrow) 6.67</td>
<td>1,131</td>
<td>971</td>
</tr>
<tr>
<td></td>
<td>3-3-2-2</td>
<td>6.67 (\rightarrow) 1,638</td>
<td>1,629</td>
<td>1,629</td>
<td>6.67 (\rightarrow) 3.33</td>
<td>4,110</td>
<td>4,040</td>
</tr>
<tr>
<td></td>
<td>3-3-3-3</td>
<td>—</td>
<td>timeout</td>
<td>timeout</td>
<td>—</td>
<td>timeout</td>
<td>timeout</td>
</tr>
</tbody>
</table>
Acknowledgments

We thank Sarah Keren for sharing with us her implementation of her GRD solvers. We would also like to thank all the reviewers for their constructive comments and suggestions. Finally, this research is partially supported by NSF grants 1345232 and 1550662. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government.

References


