DISSERTATION

HELPING HUMANS AND AGENTS AVOID UNDESIRABLE CONSEQUENCES WITH MODELS OF INTERVENTION

Submitted by

Sachini Situmini Weerawardhana

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2021

ABSTRACT

HELPING HUMANS AND AGENTS AVOID UNDESIRABLE CONSEQUENCES WITH MODELS OF INTERVENTION

When working in an unfamiliar online environment, it can be helpful to have an observer that can intervene and guide a user toward a desirable outcome while avoiding undesirable outcomes or frustration. The *Intervention Problem* is deciding when to intervene in order to help a user. The Intervention Problem is similar to, but distinct from, Plan Recognition because the observer must not only recognize the intended goals of a user but also when to intervene to help the user when necessary. In this dissertation, we formalize a family of intervention problems to address two sub-problems: (1) **The Intervention Recognition Problem**, and (2) **The Intervention Recovery Problem**.

The Intervention Recognition Problem views the environment as a state transition system where an agent (or a human user), in order to achieve a desirable outcome, executes actions that change the environment from one state to the next. Some states in the environment are undesirable and the user does not have the ability to recognize them and the intervening agent wants to help the user in the environment avoid the undesirable state. In this dissertation, we model the environment as a classical planning problem and discuss three intervention models to address the Intervention Recognition Problem. The three models address different dimensions of the Intervention Recognition Problem, specifically the actors in the environment, information hidden from the intervening agent, type of observations and noise in the observations.

The first model: **Intervention by Recognizing Actions Enabling Multiple Undesirable Consequences**, is motivated by a study where we observed how home computer users practice cyber-security and take action to unwittingly put their online safety at risk. The model is defined for an environment where three agents: the user, the attacker and the intervening agent are present. The intervening agent helps the user reach a desirable goal that is hidden from the intervening agent by recognizing critical actions that enable multiple undesirable consequences. We view the problem of recognizing critical actions as a multi-factor decision problem of three domain-independent metrics: certainty, timeliness and desirability. The three metrics simulate the trade-off between the safety and freedom of the observed agent when selecting critical actions to intervene.

The second model: **Intervention as Classical Planning**, we model scenarios where the intervening agent observes a user and a competitor attempting to achieve different goals in the same environment. A key difference in this model compared to the first model is that the intervening agent is aware of the user's desirable goal and the undesirable state. The intervening agent exploits the classical planning representation of the environment and uses automated planning to project the possible outcomes in the environment exactly and approximately. To recognize when intervention is required, the observer analyzes the plan suffixes leading to the user's desirable goal and the undesirable state and learns the differences between the plans that achieve the desirable goal and plans that achieve the undesirable state using machine learning. Similar to the first model, learning the differences between the safe and unsafe plans allows the intervening agent to balance specific actions with those that are necessary for the user to allow some freedom.

The third model: **Human-aware Intervention**, we assume that the user is a human solving a cognitively engaging planning task. When human users plan, unlike an automated planner, they do not have the ability to use heuristics to search for the best solution. They often make mistakes and spend time exploring the search space of the planning problem. The complication this adds to the Intervention Recognition Problem is that deciding to intervene by analyzing plan suffixes generated by an automated planner is no longer feasible. Using a cognitively engaging puzzle solving task (Rush Hour) we study how human users solve the puzzle as a planning task and develop the Human-aware Intervention model combining automated planning and machine learning. The intervening agent uses a domain specific feature set more appropriate for human behavior to decide in real time whether to intervene the human user.

Our experiments using the benchmark planning domains and human subject studies show that the three intervention recognition models out performs existing plan recognition algorithms in predicting when intervention is required.

Our solution to address the Intervention Recovery Problem goes beyond the typical preventative measures to help the human user recover from intervention. We propose the **Interactive Human-aware Intervention** where a human user solves a cognitively engaging planning task with the assistance of an agent that implements the Human-aware Intervention. The Interactive Human-aware Intervention is different from typical preventive measures where the agent executes actions to modify the domain such that the undesirable plan can not progress (e.g., block an action). Our approach interactively guides the human user toward the solution to the planning task by revealing information about the remaining planning task. We evaluate the Interactive Human-aware Intervention using both subjective and objective measures in a human subject study.

ACKNOWLEDGEMENTS

I would like express my deepest gratitude to Prof. Darrell Whitley, Dr. Mark Roberts and Dr. Indrajit Ray for their guidance, encouragement and insightful discussions during my long winded PhD journey. Had it not been for you, this would not have become a reality. To all the faculty members in my dissertation committee, Dr. Francisco Ortega, Dr. Sangmi Pallickara and Dr. Carol Seger, thank you very much for agreeing to be a part of this excursion and helping me succeed in more ways than one.

I fondly remember my late advisor Prof. Adele Howe; my mentor and my friend until 2017. Thank you for your courage, patience and above all for being a wonderful human being. I miss you. May you find peace.

Thank you to my husband for everything you have had to put up with in the past few years, from bringing me late night dinners to my office to listening to my incessant complaints.

Thank you to my parents for getting me this far in life. I think I can take things from here. I will try not to disappoint you.

Many thanks to all my friends in the Sri Lankan CSU student community in Fort Collins: Thilina, Waruna, Navini, Amila ayya, Vidarshana ayya, Sachintha, Shashika, Jana, Laksheen, Shamila and Miyuru. I am forever grateful to all the support you have given me throughout the years, not to mention “the entertainment” and the delicious food.

Thank you to anyone and everyone who knew that I was doing this and lent a hand in word or action. I appreciate all of you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Avoiding Undesirable Consequences

Imagine a human user learning to use a new software application. In the beginning, the user is more likely to make mistakes because he may not have all the information to work with the software. In another instance, while performing routine tasks on the computer, the user may receive a phishing email appearing to be from a well-known and trusted source, requesting personal information. The two scenarios highlight how the routine tasks performed by human users can sometimes have unintended, perhaps dangerous consequences. The same can occur when a software agent executes a task in a complex environment, where the agent may be subverted by hidden information or by a nefarious agent acting as an adversary. In the case of a human user, the cognitive load imposed by the task may be an additional impediment for the user to find a sequence of actions that completes the task safely. User error or actions of nefarious agents may cause plans to accomplish the intended goal become vulnerable to undesirable consequences. Therefore, when working in an unfamiliar environment, it is important to have a system in place to recognize in advance when the software agent or the human user is executing a plan that will result in an undesirable consequence, and upon recognition, guide the agent (or the human user) toward the goal while avoiding the undesirable consequence. We propose intervention as a utility for online assistive agents and safety critical decision making for human users. As illustrated in Figure 1.1, we model intervention as a two stage process. The environment is modeled as a state transition system. The agent (or the human user), in order to achieve a desirable outcome, executes actions $(a_1, a_2, \ldots)$ that change environment from one state to the next $(S_0, S_1, S_2, \ldots)$. For simplicity, henceforth we refer to the agent in the environment as *the user*. Some states in the environment are undesirable and the user does not have the ability to recognize them. The intervening agent, henceforth referred to as *the observer*, wants to help the user in the environment avoid the undesirable state.

In the first stage of the intervention process, which we refer to as ***the Intervention Recognition***, the observer monitors what the user doing. The observations may consist of actions and/or the

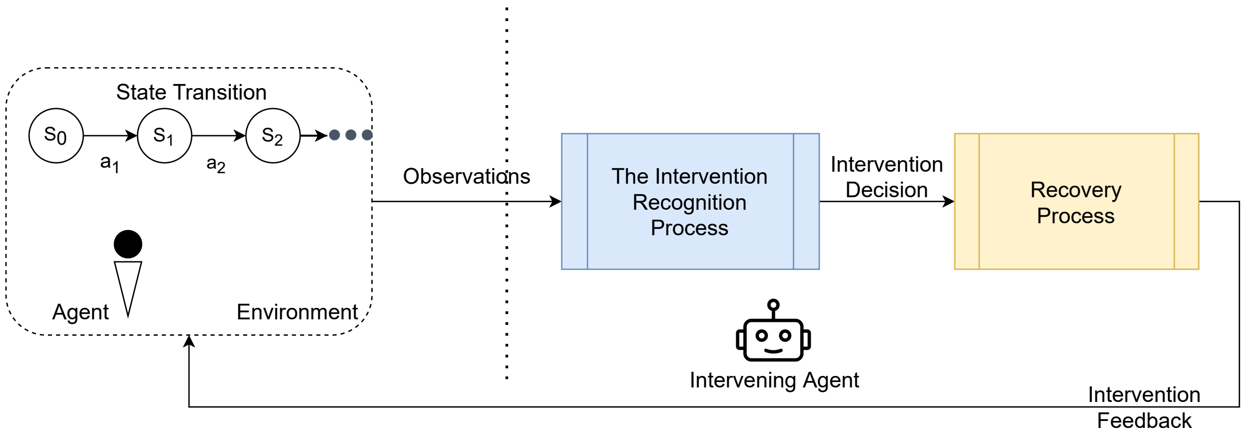


Figure 1.1: The stages of intervention

states resulting from the actions that the user executes. *Using the model of the environment and the observations as evidence, how can the observer automatically detect that an undesirable state is developing and decide when to intervene in order to help the user*? In this dissertation, we present **three intervention models** to answer this question.

The second stage of the intervention process is ***the intervention recovery process***. It is natural to envision the intervention decision to manifest as an alert message for the agent in the intervention feedback loop. The simplest case of helping the user through intervention is to adopt a preventive measure such as blocking the recognized dangerous action or displaying an alert. This approach makes sense in domains like cyber-security because once the attack is complete, reverting back to a safe state may be difficult. Furthermore, the user may not even be aware that an attack has taken place until long after. Examples of intervention actions include: cleaning attachments of a particular type, updating a software version or installing a firewall. In other domains, the preventive measures may not be as helpful. For example, if the intervention occurs when the human user is learning to use a new software application, the observer in addition to blocking the action, must also help the user in framing the decision about what to do next. Therefore, in the intervention recovery process, we ask the question: *can we improve the intervention recovery process to guide the user toward a desirable outcome while avoiding undesirable outcomes instead of blocking actions?* This is an important question, specially in cases where the agent is a human user. To answer this question, we present an **intervention feedback technique built on automated planning** to safely guide the user toward the goal. The intervention feedback technique helps a human user complete a cognitively engaging problem solving task by providing helpful hints. A hint is a piece of information about the search problem the user is solving. We design helpful hints to allow the user to carefully probe the solution space of the problem during the search, while avoiding the undesirable consequence. In this dissertation, we explore how automated planning can be used in:

- modeling the intervention environments
- intervention recognition process
- intervention recovery process

Automated planning is integral to the design of the intervening agent. First, planning is used to model the intervention environment as a state transition system comprised of preconditions and post-conditions of actions. Second, to achieve some goal in the environment, the user executes many actions; while a few of them incur risk, some are pivotal in triggering the undesirable consequence. Therefore, in order to identify when intervention is needed the observer must be able to generate alternative plans to find the many possible ways to reach the undesirable consequence. Third, when intervention occurs the user may still be incapable of deciding what to do next on his own, mainly because of hidden information about the domain or the actions of an adversary that the user can not control. Therefore, planning can be used to implement preventive measures such as blocking. Once the paths leading to an undesirable consequence have been identified, the planner can be provided with intervention actions that negates the states leading to the undesirable consequence. The preconditions of an intervention action are states that are on the path to the undesirable consequence and the post-conditions of the intervention action negates that specific state. In situations where the user needs more help than a simple preventive measure, as we will show in this dissertation, automated planning can be used to help guide the user toward a plan that avoids the undesirable consequence. Good intervention actions are those that block multiple undesirable consequences, are low cost to execute and do not interfere with the user's needs or preferences. In this dissertation, we discuss three intervention models that address two of the aforementioned requirements, blocking multiple undesirable consequences and not interfering with the user's needs.

Recognition of actions that ensures safety while allowing some freedom for the user : The observer and the user operate in an unfamiliar and online environment, where partial knowledge about the environment precludes the user from recognizing an unsafe state. The observer has knowledge about the user's desirable goal and the unsafe state the user likes to avoid. We discuss the design of an observer that can intervene and guide a user toward a desirable outcome while avoiding undesirable outcomes or frustration. When recognizing where intervention is required, the observer considers both the user's goal and the undesirable state. Furthermore, the observer needs to allow the user reach his goal and not be constantly interrupted. We present two intervention models that combine automated planning and machine learning. The observer can adopt these models to help the user reach the desirable goal while avoiding the undesirable state.

Recognition of actions that enable multiple undesirable consequences : Using a cyber-security application, we study intervention when the observer is not aware of the user's goal, but still needs to help the user avoid multiple undesirable states in the environment. Because the user's goal is unknown, the observer cannot determine when to intervene based on the user's goal and the undesirable state. Therefore, we frame the observer's decision to recognize when intervention is required as a function of objective metrics. The objective metrics capture the timeliness and criticality of actions that must be flagged for intervention in order to help the observer identify at an opportune time, the actions that may cause the most damage. We also investigate how the missing and extraneous actions in the observation trace affects the intervention decision.

In both recognition types, the observer must deal with a trade-off in early recognition of potential danger versus certainty that the undesirable effects will occur. In this dissertation, we investigate different metrics to capture that trade-off and how to combine different objective metrics to identify pivotal actions.

1.1 The Dimensions of an Intervention Problem

Let us now look at an example application that requires intervention. Figure 1.2 models an application in the Blocks World domain (Gupta & Nau, 1992), where two agents: a user and a competitor stack blocks to spell the words BAND and BRAND respectively. In the environment, the user and the competitor perform the actions PICK-UP, UNSTACK, STACK, and PUT-DOWN using the blocks A, B, N, D and R. The user can not recognize the block R. The competitor uses the hidden block R to subvert the user's goal BAND and achieve his goal BRAND. We refer to the user's goal BAND as the desirable goal (d) and the competitor's goal BRAND as the undesirable state (u). When the competitor and the user present actions to the observer, the observer must help the user achieve BAND while avoiding BRAND. The observer does this by accepting actions that help the user advance toward the goal BAND and rejecting the actions that do not. In this example, when the user first presents the action UNSTACK A B, the observer asks the question: "*will the user avoid the undesirable state if the presented action* UNSTACK A B *is accepted as an observation*?" If the observer's analysis finds that the user will not avoid the undesirable state, then intervention occurs and the process moves on to the intervention recovery phase. In the intervention recovery phase, the observer accepts actions that help the user advance toward d and rejects actions that enables or satisfies the undesirable state. The actors continue to present actions until the user achieves d. The observer makes the intervention decisions in favor of the user for each presented action.

We study several properties in the intervention environment in order to discuss the dimensions of an Intervention Problem, taking a cyber-security application and a cognitively engaging puzzle solving task as examples.

Actors in the environment : The observer may be monitoring the user working alone or the user working with other agents in the domain. The example in Figure 1.2 shows a scenario where the observer monitors two actors. The intervention models we discuss in this dissertation consider both single agent and multi-agent intervention. When there are other agents working to achieve goals that are different to the user's, the observer must analyze how the post

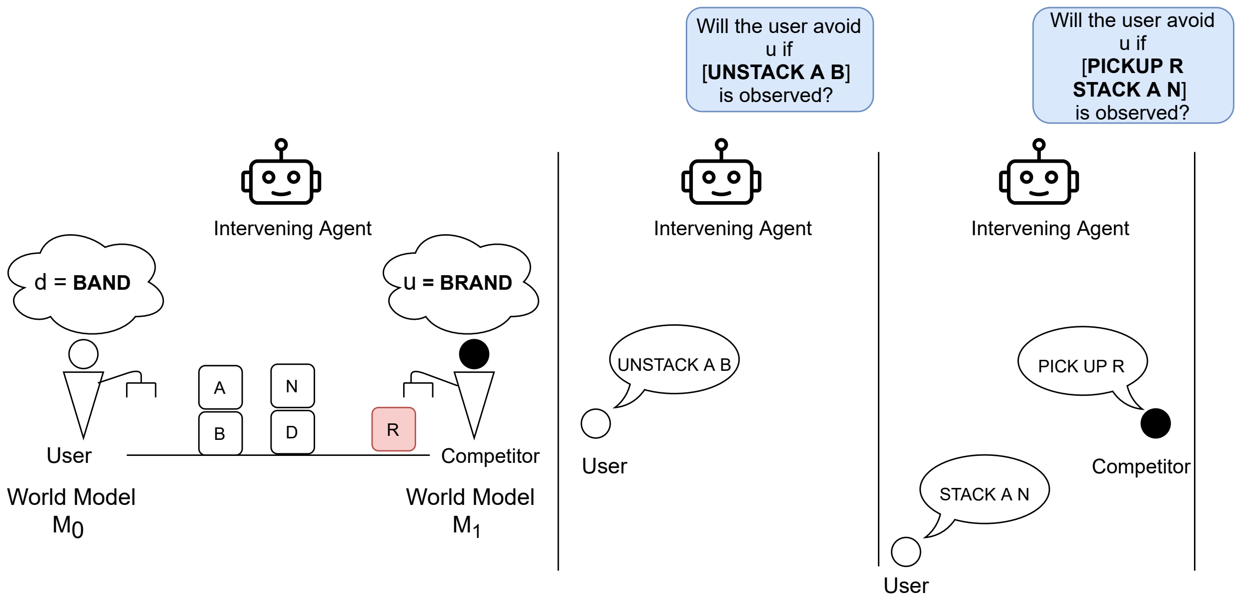


Figure 1.2: An intervention episode modeled in the Blocks Words domain

conditions of the other agents' actions affect the user's goal achievement. We can not assume that only the other agent's actions will always result in the undesirable state. Sometimes, the other agents may execute actions that enable the undesirable state, but the undesirable state does not actually manifest until much later, possibly when the user executes an action that appear to be harmless. For example, consider a phishing attack. The attacker enables the attack vector by sending the phishing email to steal the user's password. However, the user's password does not get stolen until the user clicks on the link (normally, a harmless action) to visit the phishing site and submits the information. Recognizing these enabling actions allows the observer to intervene the user in advance and give the user some time to recover.

Goals hidden to the observer : To help the user trade-off benefits versus risks, the observer needs to know what the user thinks he/she is trying to accomplish. Using automated planning, we can categorize goals of the user and other actors in the environment in terms of actions and their post conditions. However, in certain cases, specifying the user's goal may be difficult. For example, consider a home computer user preparing a document to send as an attachment in an email, while listening to music. In this situation, the home computer user will not be able to explicitly state his goals as post conditions of actions to the observer as required by planning. At different times, the human user may change the order in which he wants to achieve the goals. For example, at the start he may want to play music in the background (as a secondary goal) while preparing the document (primary goal). However, if he hears a song he doesn't like he may want to pause the document preparation task and change the song on the music player. In this situation, the user's priorities change. Therefore, in certain cases, the observer must also be able to make the intervention decision considering only the undesirable states the user wants to avoid.

Types of observations : In the example shown in Figure 1.1, the observer makes the intervention decision based on actions presented by the actors. We make the differentiation between observations of actions and observations of state because in some intervention scenarios, the observer may not be able to observe the action itself but will be able to perceive the post conditions of the action in the environment. For example, consider the situation where a remote attacker sends a phishing email to the user. The observer may not be able to observe the action of sending the email. However, the post condition of that remote action, i.e., the presence of the phishing email in the user's inbox, can be perceived by the observer. When the actions are unobservable, the states resulting from actions can be used to extract the information necessary for the intervention decision. Our intervention models consider both the observations of actions and states to decide when to intervene.

Noise in observations : When an agent or a human user executes actions to achieve a goal, lots of extraneous actions can be thrown into the observation trace. This is because the user does not intentionally want to trigger the undesirable state. Rather, he is trying accomplish a different (and useful) goal in the domain. This means that the plans that lead to the undesirable states may not encompass the user's goal and as a result, the observations will contain not only the undesirable actions, but also the desirable actions. In addition to extraneous actions, the observer may miss some actions executed by the actors because of faulty sensors. Using the cyber-security domain as case study, we explore the impact of noise in the observation trace in making a correct intervention decision.

Intervention recovery : In this work, we model the observer as an agent who can intervene to help the user reach the desirable goal safely. We present two intervention models with different intervention recovery processes. In one form of intervention, the observer considers the remaining plans in safe and unsafe partitions to learn to recognize unsafe suffixes in order to help the user avoid the undesirable state. In this case, the observer can help the user by only accepting actions into the observation history that will safely advance the user toward the user's desirable goal. In the second form of intervention, the observer learns to recognize that the user is not making progress toward d by analyzing the observation history when a suffix is not available. In this case, the observer must offer enough help to *guide* the user toward d without giving the solution. The second intervention model is particularly useful when the user is a human and it can be difficult evaluate progress with heuristics like a normal planning agent.

1.2 Intervention Use Cases

Users perform many actions in order to complete a task; while a few of them incur much risk, some are pivotal in triggering vulnerabilities. Given there are necessary preconditions in the system, either created by the user or by an adversary, any seemingly harmless action can suddenly become dangerous. The Intervention models allow us to automatically detect such vulnerable situations. We discuss helpful intervention using two use cases. The uses cases we selected to study intervention have different dimensions. Table 1.1 summarizes the intervention dimensions for each domain.

1.2.1 Intervention in Cyber-security

In the cyber-security domain, we model an attacker attempting to trick the user into compromising his security/privacy during day-to-day computing tasks. The attacker creates opportunities

Table 1.1: Dimensions of intervention use cases

Dimension	**Domain**	
	Cyber-security	Rush Hour
Actors in the environment	User, Attacker, Observer	User, Observer
Goals hidden to the observer	User's goal is hidden Multiple known undesirable states	User's goal is not hidden One or more known undesirable states
Types of observations	The observer observes user's actions The observer only observes the effects of the attacker's actions	The observer observes user's actions
Noise in observations	Has missing and extraneous observations	None
Intervention recovery	block action	Offer helpful hint

for phishing and malware attacks by making them to appear as common harmless tasks such as email and installing software. Unable to recognize these attacks in advance, the user becomes an unwitting accomplice to security breaches. The Intervention Problem for the cyber-security domain consists of three actors: the human user, the attacker (human/machine) and the observer. In most cyber-security cases, the attacker does not operate as a typical adversary (e.g., taking turns with the user). Instead the attacker takes preemptive actions and lays traps (e.g., send phishing email), which are often tied to typical actions the user performs (e.g., checking email). We model the cyber-security domain with multiple undesirable states: phishing attacks, malware installations. However, the user's goal is hidden to the observer. In the cyber-security domain we model noisy observations in terms of missing and extraneous actions. The observer considers the actions of the user and the post-conditions of attacker's actions to produce the intervention decision. The observation trace will contain varying noise levels from missing and extraneous actions. Our helpful intervention solution for cyber-security is based on automated planning and uses the properties of a planning problem to identify "*critical trigger actions*", that will cause the security breach. We do not study a specific recovery process for Cyber-security domain. We assume that when the observer identifies the critical trigger action, it is blocked.

1.2.2 Intervention in Rush Hour

Our second case study is a puzzle solving task called Rush Hour, which simulates a parking lot. In the standard version of the puzzle, the player has to clear a path on a board to move a target vehicle to the exit. To simulate the condition where the user is working only with partial knowledge about the domain, we introduced a hidden forbidden vehicle to the Rush Hour puzzle, which if moved will cause the undesirable state. We use the Rush Hour puzzle to approximate the case where human users learn to use a new software application and also it helps motivate the users to actually do the task in experiment conditions. The Rush Hour domain only has the human user and the observer. Unlike the cyber-security use case, the observer is aware of the desirable goal the user is trying to achieve (i.e., solve the puzzle by moving the target vehicle to the exit) and also the undesirable state (i.e., forbidden vehicle moves). Similar to the cyber-security use case, the Rush Hour domain also contains multiple undesirable states. All states resulting from legal moves of the forbidden vehicle are undesirable. The observations are actions the user executes to solve the puzzle. In this case, the observations are not noisy. Our intervention solution for this case uses machine learning algorithms to recognize when the user is about to move the forbidden vehicle and intervene. As the recovery process, we explore how to help the human user decide what to do next by offering further assistance as helpful hints. In both cases the user actively pursues his own goal. The threats are triggered because the user did not understand the consequences of reaching the undesirable state (phishing/malware vulnerabilities) or was not aware of the undesirable state from the start (hidden forbidden vehicle in Rush Hour). The observer makes the decision to intervene upon recognizing actions in the observations that enable or satisfy the undesirable state.

1.3 Distinguishing Intervention From Plan Recognition

Plan recognition is the closest to our Intervention Problem. In the literature the plan recognition is defined as the problem of taking as input a sequence of actions performed by an actor and inferring the goal pursued by the actor and also organizing the sequence as a plan structure (Kautz & Allen, 1986; Schmidt, Sridharan, & Goodson, 1978a). The solution to the plan recognition problem is given by the set of goals that produces a plan that is compatible with the observations. The online version of the plan recognition problem uses observations as they happen as input. The offline version requires the complete observation to be available in advance. At first glance, it might seem that intervention is a variant of Plan Recognition for the user's desirable goal and the undesirable states the user wants to avoid. However, there are several subtleties that make intervention unique, which we now discuss.

- **Intervention is an online problem.** In most cases Plan Recognition (e.g., (Ramírez & Geffner, 2009, 2010; Sohrabi, Riabov, & Udrea, 2016)) is an offline problem; there are a few notable exceptions (Mirsky, Stern, Gal, & Kalech, 2018). However, intervention is inherently online and dynamic. The observer decides whether to intervene (or not) every time the user(s) presents an action. In order to make the decision, the observer uses the observation history, which contains accepted actions. Intervention is a multi-agent problem as well. This has ramifications in environments where the user and the competitor compete to achieve close but different goals. With intervention, the observer can help the user by accepting actions into the observation history that only help further the user's goal. This is not possible with offline plan recognition.

- **Agents may have distinct views of the problem.** The user and the competitor are modeled with different domain definitions. When the agents have distinct views of the problem and they can not satisfy the undesirable state on their own, conditions will arise such that the agents working together will enable the undesirable state. The user wants to satisfy the desirable goal state, while avoiding the *hidden* undesirable state. The competitor is trying to subvert the user's goal by enabling preconditions for the undesirable state without the user's knowledge. When the user and the competitor reveal their plan(s) incrementally, the observer needs to decide whether the revealed actions make it impossible for the user to avoid the undesirable state considering the plans from the user's and the competitor's domain definitions collectively. Any action that make it impossible for the user to avoid the undesirable state must be flagged for intervention and not accepted into the observation history. If the plans for the undesirable state and desirable goal share a long common prefix, it will be difficult for the observer to disambiguate between the goals and to use plan recognition algorithms in time to help the user avoid the undesirable state.

- **Partitioned suffixes (with known desirable goal).** When the desirable goal is known, the observer should allow the user to pursue suffixes leading to the desirable goal and intervene when actions are presented from suffixes that get "too close" to the undesirable state. Our key insight is to model the "goals" of the user and competitor, which justifies our use of planning to find these suffixes. However, intervention adds new concerns beyond plan recognition, namely that the observer needs to consider two kinds of goals that might require intervention:

  1. Cases where the user is headed toward a undesirable outcome. This can be easily be solved by plan recognition.

  2. Cases where the user unwittingly enables an undesirable outcome by taking actions toward a desired goal. This is less easily solved by plan recognition because there is an inherent trade-off between intervening and allowing the user some freedom to pursue the desirable goal. Suppose that some suffix X_u leads to the undesirable state and suffix X_d leads to the desirable goal. Then it can happen that by simply following the plan leading to the desirable goal, the user enables the undesirable state when there is enough overlap between X_u and X_d.

  Partitioning the suffixes this way allows the observer to learn the differences between the safe and the unsafe suffixes and balance specific unsafe actions with those that are necessary for allowing the user some freedom. For example, in a malicious email attack such as phishing, the user will still want to check email. So prohibiting email is untenable; what the agent must do is intervene when the user attempts to click the phishing link. This need for partitioning the suffixes is distinct from plan recognition, where existing offline Plan Recognition algorithms often rely on plan cost to disambiguate between goals and recognize suffixes. When the undesirable state and the desirable goal are too close, disambiguation based on plan cost will not work (we will demonstrate in a later chapter). Learning the differences between the plan partitions will allow us to address this issue.

- **Suffix analysis with unknown desirable goal.** When the desirable goal is unknown, the observer only has the space of plans leading to the undesirable goals to make the intervention decision when the agents reveal their plans incrementally. Plan recognition over the undesirable goals does not really apply because it is focused on matching actions to prior plans to determine the goals the user is trying to achieve. In other words, the observer assumes that the user is executing some plan to reach the undesirable state. In reality, the user wants to avoid the undesirable state and reach a different (desirable) goal, which may be unknown to the observer. Therefore, when analyzing partition suffixes with known desirable goal, the observer uses the knowledge of the user's goal to determine which are critical actions that require intervention. In contrast, with unknown desirable goal, the observer analyzes the remaining undesirable suffixes to identify actions that cause the most damage to intervene the user.

- **Goal priors cannot be estimated reliably.** Plan Recognition algorithms use a prior probability distribution over the goal hypotheses to estimate the posterior probability of the likely goals given the observations. Intervention must consider the likely goals regardless of their priors. For example, the undesirable state is hidden from the user, who does not intend to achieve it (i.e., prior probability ≈ 0). In contrast, a competitor, when present, intends to achieve the undesirable state (i.e., prior probability ≈ 1). When the priors are not accurate, the observer (using Plan Recognition) may not be able to disambiguate between likely goals (and recognize the correct plan) in time to avoid the undesirable state.

- **Emphasis on intervention recovery.** In Plan Recognition, the observer uses an observation trace to derive the user's likely plan. The observation trace can be either an ordered sequence of actions (Ramírez & Geffner, 2009, 2010) or an ordered sequence of states (Sohrabi, Riabov, & Udrea, 2016). Plan Recognition does not define a method for the user to recover when the observer recognizes a plan leading to the undesirable state. With the proposed intervention models, we address that limitation with two different types of help the observer can offer to the user to decide what to do next.

1.4 Research Questions

The main objective of this research is to study different intervention models that help human users complete tasks safely. We address the following research questions:

What are the salient characteristics for deciding when to intervene? We address this question considering the dimensions of an intervention problem, summarized in Table 1.1.

Our intervention models are based on identifying actions in an observation trace that either satisfy the undesirable state or enable it. This requires the observer to evaluate the current state (i.e., state resulting from the observed actions) based on it's importance to causing the undesirable state considering the user's desirable goal (if available) and the undesirable state. In the two intervention use cases, criticality of the current state is evaluated using two different methods. For the cyber-security domain we identify salient features by analyzing the plan space. We model the cyber-security domain as a planning problem and use an existing automated planner to sample the possible solutions (plans) that project possible undesirable outcomes and then trace back to actions critical to their occurrence. The resulting plans are analyzed to extract objective metrics to predict if the current action is likely to trigger the undesirable consequences. Raising a warning to the user involves a trade-off in early recognition of potential danger versus certainty that the effects will occur. We investigate different objective metrics to capture that trade-off and how to combine these objective metrics to identify pivotal actions.

To study intervention in the Rush Hour domain, we formalize a family of Intervention Problems and show how these problems can be solved using a combination of Plan Recognition methods and classification techniques to decide whether to intervene. We characterize the observer's decision space as an Intervention Graph and construct it using an "Intervention as Classical Planning" approach to generate potential suffixes of partially executed plans. We extract domain-independent features from this graph and extend several Plan Recognition benchmarks to evaluate this approach. We then generalize these results to Human-Aware Intervention, where the observer must decide in real time whether to intervene for human users solving a cognitively engaging puzzle. Using a revised feature set more appropriate to human behavior, we produce a learned model to recognize when a human user is about to trigger an undesirable outcome.

How should information be displayed to effectively inform users following intervention? Using the Rush Hour domain, we study how an intervention model can be extended to help users continue a cognitively engaging task by providing helpful hints. In the context of the Rush Hour puzzle solving task, a hint is a piece of information about the Rush Hour search problem. We design helpful hints to allow the user to carefully probe the search space of the Rush Hour search problem, while avoiding the undesirable state. We monitor how human users achieve planning landmarks of the Rush Hour problem when intervention is supported by hints compared to intervention without hints to evaluate the effectiveness of our intervention recovery process.

How to design tools to study intervention in activities with human user participation? To identify pivotal events, the appropriate system states and user actions must be monitored and compared to a model of actions that can lead to the undesirable state. If an action that contributes to a trigger is suspected, it should be put in the context of what the user is trying to do and an estimate of how likely the action is to actually cause the undesirable state. We develop state/action models using automated planning that allow us to recognize the appropriate system *states* (computer/puzzle board) that must be monitored and the *actions*, which may lead to the undesirable state. For the cyber-security domain, we model the security vulnerabilities that occur in a home computer environment using PDDL (Planning Domain Definition Language) (Ghallb et al., 1998), which is designed to represent pre-conditions and post-conditions of actions. PDDL is the prominent representation used in the AI Planning community. We also develop PDDL models for the Rush Hour domain, which will be used in generating the helpful hints. Our studies are focused on how human users solve these tasks (e.g., perform common home computer user actions, solve a Rush Hour puzzle), which requires studying human users in situ. To this end, we will develop a simulated home computer environment for studying users practice computer security and a Rush Hour puzzle simulator to study how human users respond to helpful hints. Both these tools are event monitoring systems that capture human user's actions when placed in different experimental conditions as well as system level events that are defined in the PDDL models.

The rest of this dissertation is organized as follows:

- **Chapter 2**: We present the background on automated planning and a review of existing plan recognition literature that studied methods to infer an agent's intention.
- **Chapter 3**: We present the findings of an human subject study to motivate plan intervention. In this study, we simulate cyber-security vulnerabilities that occur in a home computer environment when the human users perform routine tasks like checking email and using software applications.
- **Chapter 4**: We present the first intervention model based on the findings of the cyber-security human subject study. We formulate the Intervention Problem as a planning problem of three domain independent objective metrics: timeliness, which captures how soon the undesirable state may occur; certainty, which captures how frequently the undesirable state may be seen; and desirability, which captures the user's preference for continuing the current action despite the increased risk. Against an ideal baseline, we examine trade-offs in choosing the "correct" intervention point by varying the weights associated with these objective metrics, the observability of actions, and the presence of extraneous actions.
- **Chapter 5**: We study two kinds of Intervention Problems in environments where an observer monitors a user (and a competitor) and help the user achieve a desirable goal, while avoiding an undesirable state. In "**Unsafe Suffix Intervention**", the observer uses automated planners to project the remaining suffixes and extract features that can differentiate between safe and unsafe plans. We evaluate the recognition accuracy of Unsafe Suffix Intervention on benchmark planning problems. In "**Human-aware Intervention**", the observer uses the observed partial solution to extract features that can separate safe and unsafe solutions. We evaluate

the accuracy of Human-aware Intervention on a new Intervention Planning benchmark Rush Hour.

- **Chapter 6**: We present an intervention recovery process for the Human-aware Intervention model based on automated planning. We discuss the findings of a human subject study where human users solved a cognitively engaging Rush Hour puzzle task while being guided by a human-aware intervention agent. We evaluate the efficacy of our recovery approach using automated planning landmarks.
- **Chapter 7**: We discuss some open issues in Intervention and provide an outline for future research.

Chapter 2

Preliminaries

In order to design intervention for human users, first we need to model human user behavior in some environment. We use the automated planning formalism (Nau, Ghallab, & Traverso, 2004) to describe the tasks the user performs in both the cyber-security and Rush Hour domains. The automated planning formalism is based on concepts such as goals, states and actions, to describe the behavior of rational agents. Although human user behavior is more complex than that of a rational agent, in our intervention models we assume the actor being intervened is a (bounded) rational agent. In this chapter, we first present an overview of automated planning and describe how a state/action model can be represented as a classical planning problem. Next, we discuss the problem of inferring the intention of an agent and discuss three variants of the problem: plan recognition, goal recognition and activity recognition. Finally, we present a discussion on the existing literature on intention recognition considering a three-dimensional framework: number of agents in the environment, online/offline recognition and how the recognizer interacts with the environment during the intention recognition process.

2.1 Automated Planning

Planning aims at achieving some predefined objectives through a deliberation process that chooses and organizes actions by anticipating their outcomes. The planning formalism views the problem as a state transition system, defined as a tuple $\Sigma = (S, A, E, \gamma)$ where:

- $S = \{s_1, s_2, \ldots\}$ is a finite or recursively enumerable set of states;
- $A = \{a_1, a_2, \ldots\}$ is a finite or recursively enumerable set of actions;
- $E = \{e_1, e_2, \ldots\}$ is a finite or recursively enumerable set of events; and
- $\gamma : S \times (A \cup E) \mapsto 2^S$ is a state transition function.

Given an action $a \in A$ and $\gamma(s, a) \neq \emptyset$, then a is *applicable* in state s. If a is applied to the state s, it will take the system to a different state $s' \in \gamma(s, a)$. A state transition system Σ can be represented in a directed graph $G = (V_G, E_G)$, where $V_G = S$ and an edge $e_G \in E_G$ connects two states $e_G = s \mapsto s'$, labeled with an action a if and only if $s' \in \gamma(s, a)$ as shown in Figure 2.1.

In this example, we model the Blocks Words problem (Ramírez & Geffner, 2009), which is a modification of the Blocks World domain (Gupta & Nau, 1992). In the Blocks Words problem, the agent operates in an environment that contains a flat surface (a table) and a set of blocks identified by English letters. The blocks can be stacked on one another. The agent uses a set of actions (e.g., pick-up, put-down, stack, unstack) to build a stack such that it spells a word from the top to bottom (i.e., goal state). The agent can only move one block at a time. The agent's goal is to spell the word BAND. Initially, the blocks D, N are on the table and B is stacked on A. The partial graph shows some of the the state transitions that can happen in the domain. Red color edges show a path (i.e., an action sequence), which translates the initial state to the goal state. Let us map the Blocks Words environment in Figure 2.1 to the definition of Σ.

- $S = \{init, goal, s_1, s_2 \ldots\}$ is a finite or recursively enumerable set of states;
- $A = \{pickup, unstack, stack, putdown\}$ is a finite or recursively enumerable set of actions;
- $E = \{\}$ assuming no exogenous events occur and;
- γ : transitions indicated by arrows in the graph.

The state transition system Σ describes all the ways in which our environment may evolve. Finding a *plan* means that we need to extract a structure that gives appropriate actions to apply such that some *objective* can be achieved. There can be many forms to define this objective: (1) defining a goal state or a set of goal states (e.g., state *Goal* in Figure 2.1), (2) optimize for a utility function attached to the states, (3) satisfying some condition over the sequence of states, (4) defining tasks to be performed. In our work, when we refer to the *objective*, we mean type (1) objective.

The preliminary intervention designs are for an agent whose behavior can be represented using a classical planning model. The classical planning formalism requires eight restrictive assumptions about Σ:

1. The system is defined using a **finite** sets of states, actions and events.
2. The system defined in Σ is **fully observable**. This means that the agent has knowledge about every aspect of the state when an observation is made.
3. The system defined in Σ is **deterministic**, i.e., an action in the system does not have alternative outcomes. Formally, for all $s \in S, u \in A \cup E; |\gamma(s, u)| \leq 1$
4. The system define in Σ is **static**, i.e., there are no exogenous events happening in the environment. $E = \emptyset$
5. Only **restricted goals** can be given to the planner. Restricted goals are given as an explicit goal state or a set of goal states.
6. The solution plan for a planning problem in Σ is a linearly ordered finite sequence of actions.
7. The system defined in Σ has **implicit time**. This means that actions and events do not have any time duration between state transitions.
8. Planning in Σ is **offline**, which means that any changes to Σ that takes place while the planner is searching for a plan, are ignored.

Given a description of Σ, the initial state and an objective, an automated planner generates a plan that achieves the objective. A classical plan for the Blocks Words problem is a sequence of actions (e.g., `unstack B A, put down B, pick up N`, etc.). A plan can also be represented as a policy: a partial function from S into A (e.g., $\{$(`init, unstack B`), (s_1`, putdown B`), (s_4`, pickup N`) $\ldots\}$. When we refer to plans in this work, we mean a sequence of actions. The agent can execute these plans on the environment using actuators, which generates observations. In our Blocks Words example, the agent can use a mechanical arm to lift block B from the top of A. This generates an observation, the mechanical arm holding block B and the top of block A now being clear. Formally, the agent's behavior can be defined as a **planning problem** $P = (\Sigma, s_0, S_g)$ where,

- $\Sigma = (S, A, \gamma)$ is a state transition system,
- $s_0 \in S$ is the initial state and
- $S_g \subset S$ is a set of goal states.

The solution to P is a sequence of actions $\langle a_1, a_2, a_3, \ldots, a_n \rangle$ such that it corresponds to a sequence of states $\langle s_0, s_1, s_2, \ldots, s_n \rangle$ where $s_1 = \gamma(s_0, a_1), s_2 = \gamma(s_1, a_2), \ldots, s_n = \gamma(s_{n-1}, a_n)$ and $s_n \in S_g$

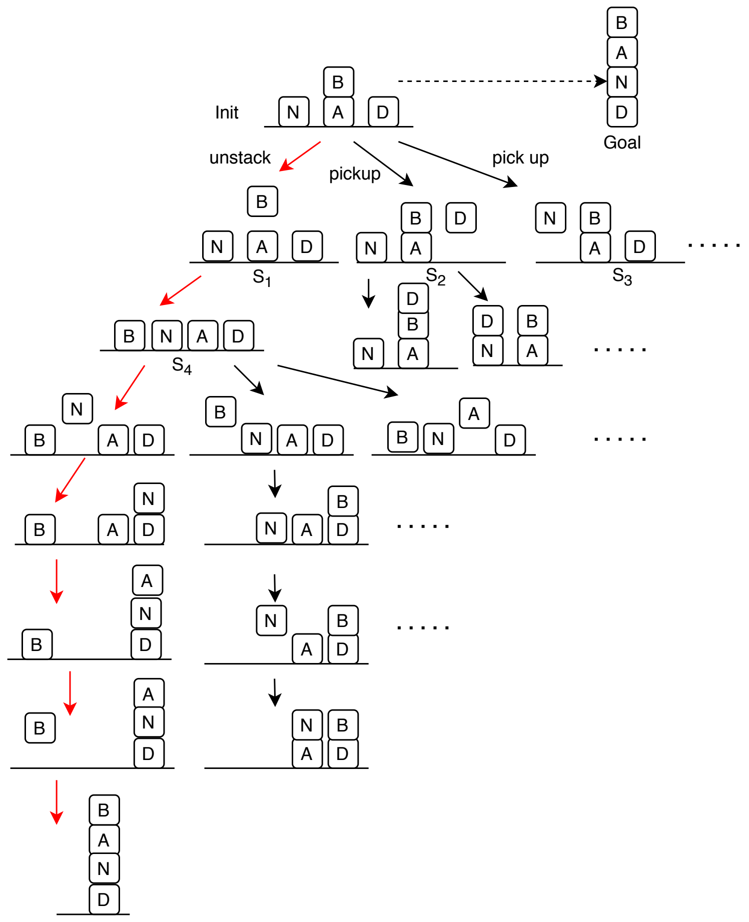


Figure 2.1: Blocks Words problem as a state transition system

2.2 Representing Classical Planning Tasks

Although we can use the graph representation of the state transition system corresponding to a planning problem, the resulting graph can be very large and modifying this graph as new events occur in the system can become very cumbersome. In order to provide a compact representation of the transition system, *states* (i.e., vertices) are represented using state variables, and actions (i.e., edges) are represented as *operators* with preconditions and post-conditions specified as state variables. This way, in order to find a plan, we only need to provide the initial and goal states and use the operators to find other states as needed.

A classical planning problem can be represented using the **classical representation**, **the set-theoretic representation** or **the state-variable representation**. STRIPS (Fikes & Nilsson, 1971), which we will use to model the domains described in this work, is an implementation of the classical representation. STRIPS uses first-order literals to describe states and actions. The set-theoretic representation is used when planning problems are solved with SAT solvers. The state-variable representation is implemented in SAS formalism (Backstrom & Klein, 1991).

In the classical representation, the world state is a set of **grounded** propositional state variables. Σ is described using a finite set of grounded propositional state variables, which means there are only finitely many possible states. Operators that modify the world state consist of precondition propositions: propositions that must be true for the action to execute, and post-condition propositions: those will be made true as a result of the action. An *action* is a ground instance of an operator. The initial state in the Blocks Words example in Figure 2.1 described in the grounded classical representation is: { `ontable(N)`, `ontable(A)`, `ontable(D)`, `on(B,A)`, `clear(N)`, `clear(B)`, `clear(D)` }. The operator unstack is described as:

operator: `unstack (x, y)`
preconditions: `(on x, y),(clear x),(handempty),¬(equals x y)`
effects: `(holding x),(clear y),¬(clear x),¬(handempty),¬(on x, y)`

Grounding the unstack operator using blocks B and A defines the action as follows:

action: `unstack (B, A)`
preconditions: `(on B, A),(clear B),(handempty),¬(equals B A)`

effects: `(holding B),(clear A),¬(clear B),¬(handempty),¬(on B, A)`

An action is *applicable* in state s if s satisfies the preconditions of the action. This means that the action's positive preconditions are in s whereas the negative preconditions are not in s. In the Blocks World example, (`unstack B A`), (`pickup D`), and (`pickup N`) are applicable in the initial state. If an action is applicable to s, the result of performing the action means to delete the negative propositions from the state s and add the positive ones. For example, if `unstack B A` is executed in the initial state, the resulting state would be { `ontable(N)`, `ontable(A)`, `ontable(D)`, `¬on(B,A)`, `clear(N)`, `¬clear(B)`, `clear(D)`, ¬(`handempty`), `clear (A)`, `holding (B)` }. Typically, in the classical representation, the state only explicitly contains the true propositional variables. Any propositional variable that are not included in the state is considered false.

2.2.1 STRIPS Planning Task

We now present a formal definition of a STRIPS domain and planning problem. In STRIPS, a planning task is defined using a planning domain and a planning problem.

Definition 1. *A STRIPS planning domain is a tuple $\mathcal{D} = \langle F, Op \rangle$, where F is a finite set of state variables (predicates) and Op is the set of operator schema. An operator schema $op \in Op$ is a tuple $op = \langle Pre(op), Add(op), Del(op) \rangle$ that consists of preconditions, add and delete effects respectively, where $Pre(op)$, $Add(op)$, $Del(op)$ are all subsets of F. Predicates and operator schema have parameter lists and can be instantiated with objects (defined later). An instantiated operator (action) op is applicable in a state s if predicates in $Pre(op)$ are True in s. A state transition induced by an action op in a state s is defined as a function δ:*

$$\delta(s, op) = \begin{cases} s \setminus Del(op) \cup Add(op) & Pre(op) \subseteq s \\ undefined & otherwise \end{cases}$$

Definition 2. *A STRIPS planning problem is a tuple $P = \langle \mathcal{O}, s_0, G \rangle$, where $\mathcal{O}$ is the set of objects. Objects can be either constant or non-constant and may have a type. Constant objects are common* *to all instances of the domain definition and shared by planning problems defined on that specific planning domain. Non-constant objects are unique to a specific planning problem. $s_0 \subseteq F$ is the set of propositions that are true in the initial state, $G \subseteq F$ represents the goal specification.*

Given a domain $\mathcal{D}$ and a planning problem P, an automated planner generates the planning task $\Pi = \langle \mathcal{F}, \mathcal{A}, s_0, G \rangle$, where $\mathcal{F}$ is a finite set of grounded propositions, $\mathcal{A}$ is the finite set of grounded actions instantiated from the operator schemata Op, s_0 is the grounded propositions specifying the initial state and G is the grounded goal specification. Objects in $\mathcal{O}$ are used to ground $\mathcal{F}$, $\mathcal{A}$, s_0 and G. Grounding replaces the variable terms in the parameter lists of the propositions, action schema, goal specifications with constant/non-constant objects.

Definition 3. *A solution to Π is a plan $\pi = \{a_1, \ldots, a_k\}$ of length k that modifies s_0 into G by successive execution of actions $a_1, \ldots, a_k$*

Actions in π may have a cost that assigns a non-negative value to the action schema defined in the STRIPS domain definition. The cost function is given as $C : Op \rightarrow \mathbb{R}^0_+$. The cost of the plan $c(\pi)$ is $\sum(c(a_i))$. The optimal solution, the optimal plan π^*, minimizes the cost.

2.3 The Recognition Task

Besides being able to plan, in certain cases the agent must also be able to *recognize* another agent's behavior. This is specially important in the domains we are studying in this dissertation where the agent acts as an assistant to the human user.

We now extend our discussion to a multi-agent setting where, we consider at least two agents: an acting agent A and an observer agent R. To draw a realistic example, consider the Blocks World scenario in Figure 2.2. Agent A solves a planning task $\Pi_A = \langle \mathcal{F}_A, \mathcal{A}_A, s_{I_A}, G_A \rangle$ executes a sequence of actions $O = \{\texttt{unstack D A}, \texttt{putdown D}, \texttt{unstack R P}\}$. For simplicity let us assume that R does not execute any actions, and passively observes the actions A executes.
In the recognition task, R asks the question: "What is A trying to do?". This question can be answered in three ways, leading to three types of recognition tasks:

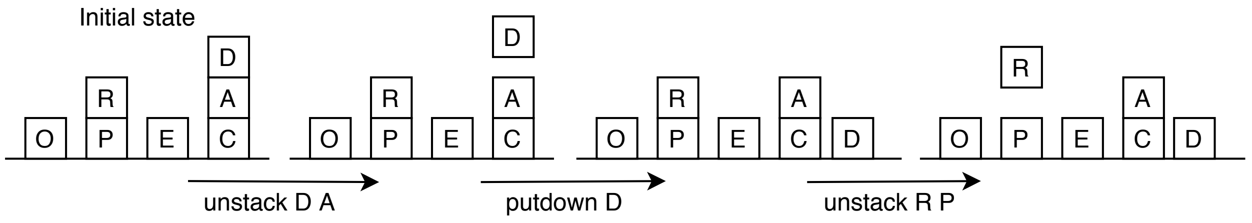


Figure 2.2: A recognition task

- Given O, what is the posterior probability P(π|O) of the complete plan π. This is the **plan recognition problem**.
- Given O, what is the posterior probability P(G|O) of the goal G, where $G \in \mathcal{G}$, a list of potential goals in the environment. This is the **goal recognition problem**
- Given O, what is the posterior probability P(a|O) of an activity a. This is the **activity recognition problem**.

Typically, R's objective is to find the most likely plan, goal, activity from a set of potential plans, goals, activities in the environment. A recognition task consists of three components:

The environment $E = \{\Sigma, s_0, \mathcal{G}\}$, is the setting in which agent A acts. s_0 is the initial state and $\mathcal{G}$ is the set of possible goals. In order to perform the recognition task, the environment may produce a **plan** $\pi = \{a_0, a_1, \ldots, a_n\}$: a complete sequence of actions that takes agent A from s_0 to the goal state G in $\mathcal{G}$, a **history** $h = \{s_0, s_1, \ldots, s_n\}$: a sequence of states from s_0 to G or an **execution** $e = \{s_0, a_0, s_1, a_1, \ldots, s_n, a_n, s_{n+1}\}$: a sequence of state-action transitions from s_0 to G. These are referred to as *prefixes* in recognition. The recognition task example shown in Figure 2.2 considers the plan prefix. Some work in the plan recognition literature uses plan prefixes while others use the history prefixes. We will discuss work on each category. Furthermore, research has investigated recognition in both discrete and continuous domains where actions transition from one state to another via paths through the state space, rather than through discrete states (e.g., navigation domains).

The acting agent (agent A) must specify the assumptions made with regard to how agent A acts in the environment to achieve his goal. Most of the time, recognition tasks assume that agent A enters the environment and follows a plan to achieve some goal. However, A's behavior may be affected by how familiar it is with the environment (e.g., are there objects that escape agent A's sensors?), A's special capabilities (e.g., can he compute an optimal plan to reach G), and the relationship to the recognizer (e.g., does A acts in such a way that his goal/plan/activity can be recognized easily or would he act to obfuscate his true goal/plan/activity?). There could be more than one agent whose goals/plans/activities we would want to recognize. The recognition literature uses different ways to represent agent A from the point of view of R in the situations mentioned above. Commonly used representations are **plan libraries** and **domain theories**.

The recognition system (agent R) When defining a recognition problem, the final component is the specification of the recognizer. Here, we need to define several components:

- the observability: how does agent R, perceive agent A's behavior (e.g., is there any noise in the observations? Are there any actions/states R can not perceive?)
- the objective: what does R need to do? (e.g., recognize A's goal/plan/activity as soon as possible? is the recognition online or offline?)
- the possible interventions: can R interact with A or affect its behavior? (e.g., provoke A to change its course during execution, modify Σ to help recognition)

We now present a framework, within which we discuss selected prior work from goal and plan recognition. We set up the framework along three dimensions:

single vs. multi actor In the single actor case, R makes observations of a single actor agent to recognize the goal/plan of that agent. In the multi-agent case, the observations originate from many agents. Furthermore, the goals pursued by the agents in the system may be a complex goal. For example consider an agent X assisting two other agents (Y and Z) pursuing two different goals G_Y, G_Z respectively. X's goal G_X an be defined as $G_Y \cup G_Z$. Thus, in a multi-agent system R may have to recognize complex goals as above, instead of one goal shared by all agents.

online vs. offline In online recognition, the observations are incrementally revealed to R. Recognition is performed based on the observations revealed thus far (i.e., a partial observation trace). In offline recognition, the complete observation trace is available to R up front and recognition is performed accounting for information available from the full observation trace.

the recognizer's interaction with the actor In this dimension we discuss different ways R can communicate with A during the recognition process. In most related work, the recognition task terminates when R identifies what A's goal or plan is. Interaction is an extension of the typical recognition problem. Some work on interaction suggests modifying the environment (e.g., blocking some actions A could otherwise have executed) to facilitate early recognition of A's goals/plans. Others propose question-answering approach where R queries A about his plans/goals and reasoning about information acquired from the queries.

Within the three-dimensional framework, for each related work shown in Table 2.1 we also discuss the properties of the **environment**, the **acting** agent and the **recognizer** agent around which the recognition tasks are modeled. Our discussion on related work also extends to work that are not listed in Table 2.1. We do this mainly to draw comparisons between how the work listed in Table 2.1 improves/extends other (earlier) noteworthy plan and goal recognition work.

2.4 Plan Recognition

The plan recognition problem is to *take as input a sequence of actions performed by an actor and to infer the goal pursued by the actor and also to organize the action sequnce in terms of a plan structure* (Schmidt, Sridharan, & Goodson, 1978b). This requires some method to represent how the actor would act in the environment. Recognition solutions that use automated planning represent the actor's behavior as a planning problem focusing on the fact that the actor moves from

Table 2.1: Fitting related work to the analysis framework

Related Work	Plan Recognition			Goal Recognition		
	Single/Multi	Online/Offline	Interaction	Single/Multi	Online/Offline	Interaction
Ramirez et al. 2009				Single	Offline	None
Ramirez et al. 2010	Single	Offline	None			
Sohrabi et al. 2016	Single	Offline	None			
Vattam et al. 2015	Single	Offline	None			
Vered et al. 2017				Single	Online	None
Boddy et al. 2005	Single	Offline	None			
Keren et al. 2014				Single	Offline	Yes
Pozanco et al. 2018				Muti	Online	Yes
Mirsky et al. 2016				Single	Online	Yes
Shvo et al. 2018	Multi	Offline	None			

state to state and changes the environment to achieve some goal. The solutions that use parsing assume that plans are constructed as a hierarchy in the actor's mind.

Early solutions to the plan recognition problem require that the part of a plan given as input to the recognizer be matched to a repository of example plans. This repository is referred to as a *plan library*. The plan library represents possible plans the acting agent may be trying to execute in the environment E. In the plan recognition literature, several methods have been proposed as representation models of the plan library.

2.4.1 Plan Recognition with Plan Library Representations

Many plan library representations use graphs to represent a possible plan that will likely execute in the environment. In The Generalized Plan Recognition (Kautz & Allen, 1986), the recognition problem is defined as identifying a *minimal set of top-level actions sufficient to explain the set of observed actions*. Here, the plans are modeled in a graph. The nodes of the graph represent the actions. The graph has two types of edges: action specialization (between top-level actions) and action decomposition into sub-actions. The authors use a cooking domain as the example and define the top level actions and recursively decompose those actions until no further specializations can be made. Lower level actions in the plan graph *logically imply* the higher level actions. For example the top level action "PrepareMeal" specialized into "MakePastaDish", specialized into "MakeFettuciniMarinara", decomposed into two leaf-level actions "MakeFettucini" and "Make-

Marinara". The implication relationships are derived from bottom to top. Given this representation of the plan library, the plan recognition task becomes the minimum vetex cover problem of the graph. This recognition problem contains one actor and one recognizer.

Geib and Goldman represent the plan library for a cyber security domain as partially ordered AND/OR trees (C. Geib & Goldman, 2009). Their representation also assumes that the tasks the actor agent is trying to complete in the domain are hierarchical. AND nodes in the graph require that all sub-tasks be achieved before completing the parent task. OR nodes represent instances where the actor agent may choose between alternatives to complete the parent task. The subtasks themselves may further be represented as AND/OR (sub) trees. The root of the tree is the goal. Plans are derived from traversing to leaves from the root of the tree. The plan library is augmented with probabilities (i.e. prior probabilities of root goals, OR tree choice probabilities etc.). Then, in order to recognize the actor's plan, the recognizer needs to compute the probability of an explanation (plan) given the observations (i.e. P($\pi|O$)). Specifically, the recognizer needs to derive a probability distribution over a set of likely explanation plans. To extract the likely explanation plans, the authors define a grammar to parse the AND/OR trees and generatively build the explanations by starting off with a "guess" and refining it as more observations arrive. The actor in this work is hostile. This means that the actor does not want the recognizer to find out his plan and he will hide some actions. This causes the recognizer to deal with partial observability when computing P($\pi|O$). This topic is left for future work.

Plan Libraries: What is the Problem Anyway?

One issue with using plan libraries is that it is difficult to ensure the completeness of the plan library. It is unrealistic to encode *all* possible plans for a given planning problem into a plan library. However, the system must still be able to reason about new but correct plans the agent may pursue. Cohen et al. propose a new approach for updating a plan library with new plans (Cohen & Spencer, 1993).

Another issue with using plan libraries for recognition is the noise in the input observations. A key assumption in the library based solution is that the observations must be able to associate with one or more recipes in the library, and the partial plan inferred for the agent must reflect the agent's intended goal or plan. Noise in the observation trace may occur from missing actions (e.g., noisy sensors, agent deliberately hiding observations, etc.) or exogenous events that occur in the environment that have nothing to do with the agent's true goal/plan. Rich et al. proposes a method to enable the plan library to incrementally relax the constraints on recipes before declaring that the recipe is not found (Rich, Sidner, & Lesh, 2001). They use a *focus stack*, that maintains a set of annotated goals that are being pursued. The goals are removed from the tack only when the goal is complete.

Vattam et al. 2015 - Case-based Plan Recognition

The case-based plan recognition (Vattam & Aha, 2015) relaxes the error-free requirement of the observations in plan recognition problems and introduces a recognition algorithm that can handle **missing** and **misclassified** actions in the observation trace. This problem is also modeled with a single actor and a single passive recognizer agent and assumes that there exists a plan library consisting of a set of *cases*. A case consists of a planning problem and a fully grounded solution plan to this planning problem. Given a subsequence of actions, the algorithm needs to address two requirements: (1) matching the incoming sequence to the stored cases to retrieve candidate plans and (2) evaluating the retrieved cases to identify the top-ranked candidate plan (i.e., recognize plan).

Cases are stored using labeled, directed graph called an *action sequence graph*. This data structure facilitates comparisons between incoming observations and the stored plans using similarity metrics derived from the graph topology. In order to account for errors in the observations, plans are represented as a sequence of action-state pairs: $\mathbb{S} = \langle (null, s_0), (a_1, s_1), \ldots, (a_g, s_g) \rangle$. s_0 is the initial state and s_g satisfies the goal g. This is in contrast to the typical representation of a plan, which is a sequence of actions. Given an action-state pair $(a, s)_k \in \mathbb{S}$, they first produce the predicate encoding graph ϵ, a labeled, directed graph such that the vertices encode the action a, the state fact s and the corresponding parameters of a and s. Then two types of edges are created for both action and the state fact that connect the predicate and the first parameter and an edge for each pair of parameters in the predicate's parameter list. An edge is labeled based on the two nodes that it connects. Once each action-state pair $\in \mathbb{S}$ generates its own ϵ, the action sequence graph is generated by their union.

Given the action sequence graph of a sequence of action-state pairs and a plan library consisting of action sequence graphs, candidate plans are retrieved by evaluating the structural similarity between them. This work proposes two graph based similarity metrics (degree sequences similarity metric, combined similarity metric), which can approximate the *maximum common subgraph* to match the incoming graph to the graphs in the library. Then the top k graphs based on the approximated metric values are selected as the recognized plans.

Constraints in the Approach

This approach is tested on the Blocksworld benchmark domain. The plan library is small and errors to the observation sequences are introduced systematically, rather than attempting to simulate realistic conditions. In the application domains we are studying for this research, the human users interact with the recognition system, which means that errors may appear in the observation trace in an ad-hoc manner. Furthermore, the user (due to partial visibility/lack of knowledge) may not realize the action they take is incorrect and do it anyway. This requires that the recognition system be able to "evaluate" partial observation sequences based on the extent to which it helps task completion while avoiding errors.

Next, we introduce a second approach for solving plan recognition problem, which does not require the existence of a plan library.

2.4.2 Solving the Plan Recognition Problem with Domain Theory

Ramirez et al. 2010 - Probabilistic Plan Recognition

Ramirez and Geffner proposed a plan recognition solution that does not rely on defining a plan library (Ramírez & Geffner, 2010). Their approach has the advantage of being able to exploit automated planners to sample the plan space to find plans that are compatible with the observations.

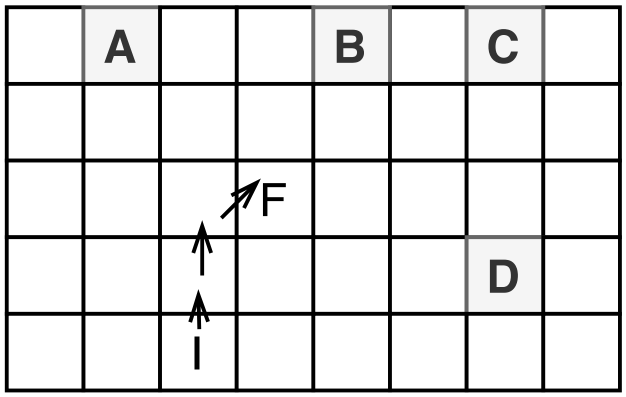


Figure 2.3: The plan recognition problem

This allows the recognition problem to scale up well to handle domains with a large number of actions and state variable predicates.

The environment E discussed in this solution consists of one actor agent and one recognizer. The actor is not executing an optimal plan. The recognizer passively observes the actions executed by the actor. Figure 2.3 illustrates a grid based plan recognition task, where an actor moves through a 8×5 grid. The actor initially is the position indicated by **I**. The actor attempts to accomplish one of the possible set of goals {A, B, C, D}, by moving either horizontally, vertically, or diagonally. Using PDDL, we can represent the possible set of goals $\mathcal{G} = \{$ `(at A)`, `(at B)`, `(at C)`, `(at D)`$\}$. **F** is the current position of the actor. Assume that horizontal and vertical moves have cost of 1, and diagonal move has a cost of $\sqrt{2}$. The arrows show the path the actor has taken, which consists of three moves (2 vertical and 1 diagonal). The recognizer asks the question, "*Given the three moves (up,up,diagonal) what is the actor's most likely goal*?" Formally, the plan recognition problem $\mathcal{R}$ is a tuple such that $\mathcal{R} = \langle \mathcal{F}, \mathcal{A}, s_0, \mathcal{G}, O, PROB \rangle$, where $\mathcal{F}$ is the set of state variable predicates, $\mathcal{A}$ is the set of actions, s_0 is the initial state, $\mathcal{G}$ is the set of possible goals, O is an observation sequence $O = \{o_1, \ldots o_m\}$ and $PROB$ is a prior probability distribution over $\mathcal{G}$. In this work, each $o_i \in O$ is an action in $\mathcal{A}$. Note that $\mathcal{F}, \mathcal{A}, s_0$ are collectively known as the *planning domain theory*.

Ramirez and Geffner provides a solution to the recognizer's question by accounting for the cost differences of two types of plans for each goal: (1) plans that reach a goal while going through the observations and (2) plans that reach a goal without going through the observations. Formally, $\forall g \in \mathcal{G} : \Delta_g = C_g(O) - C_g(\overline{\mathrm{O}})$, where Δ refers to the cost difference, $C_g(O)$ refers to the cost of the plan that reaches goal g going through O and $C_g(\overline{\mathrm{O}})$ is the cost of the plan that reaches the goal without going through O. In the example in Figure 2.3, when the agent is (`at F`),

$\Delta_{(atA)} = (2 + 3\sqrt{2}) - (3 + \sqrt{2}) = 1.8$

$\Delta_{(atB)} = (3 + 2\sqrt{2}) - (2 + 2\sqrt{2}) = 1$

$\Delta_{(atC)} = (6 + 2\sqrt{2}) - (4\sqrt{2}) = 3.2$

$\Delta_{(atD)} = (4 + 2\sqrt{2}) - (3 + \sqrt{2}) = 2.4$

This indicates that from the given observations, (`at B`) has the least cost difference. What does that mean? According to the Δ_g formula, if for a specific goal, if $C_g(O)$ is greater, it follows that $\overline{\mathrm{O}}$ is more likely than O (because the rational actor agent would follow the lower cost plan). Putting it differently, smaller Δ means that cost of $\overline{\mathrm{O}}$ plan is greater, thus the agent is likely following the plan that is aligned with the observations. Therefore, given the observations, the most likely goal would be the one that has the least cost difference. The solution to the plan recognition problem is expressed as the subset of goals $g \in \mathcal{G}$ such that the *optimal* plan for g satisfies the observation sequence O.

How does the recognizer produce plans that are compatible with the observations? For this, Ramirez and Geffner use a technique called "*Compiling Observations Away*". When the observations are compiled into the domain theory, it forces the automated planner to search for solutions that already contain the observations. They transform the original domain (defined in STRIPS) D into a domain D' given the observations O. The new domain has the same initial state and actions as D. However, new state variable predicates are added to D' such that if an action a is in O, it's definition in the domain gets added an extra state variable predicate to its post-conditions. This modification is done only if a is the first observation in O. If there is an action $b \in O$ that immediately comes before a, then the preconditions of action a gets added a new state variable predicate (that is already in the post conditions of b) and post conditions of action a gets added a second new state variable predicate.

Once these modifications are done to the domain definition D', automated planners can be used to find plans that are compatible with the observations. D is used to find plans that are not compatible with the observations. Then, to do the recognition task for each $g \in \mathcal{G}$ they compute the posterior probabilities $P(g|O) \cong \alpha P(O|g)P(g)$, where $P(g)$ is the prior probability for g given as input to the recognition task and α is the normalizing constant. Ramirez and Geffner characterize the likelihood $P(O|g)$ as a Boltzmann distribution $P(O|g) = \frac{e^{-\beta\Delta_g}}{1+e^{-\beta\Delta_g}}$ where β is a positive constant and Δ_g is the cost difference between observation compatible and not compatible plans for goal g.

Constraints in the Approach

In order to compute the posterior probabilities over the likely goals, the planner has to be invoked twice: once to find the observation compatible plans and once to find the plans that are not compatible. This may be costly for large problems and plan recognition in real time (i.e., when observations arrive incrementally). Furthermore, for the two costs to be different (and enable the recognizer to correctly identify the actor's goal) the observations must contain action landmarks for the goal we want to identify. Landmarks are state predicates that must be true in any valid plan to reach the goal (Hoffmann, Porteous, & Sebastia, 2004). Action landmarks contain the landmarks as their post-conditions. If the observed action is an action landmark for g, then $C_g(\overline{\mathrm{O}})$ will be higher and as a result $P(g|O)$ will be higher. If there are many ways to reach g (i.e., no action landmarks) then the cost difference $C_g(O) - C_g(\overline{\mathrm{O}})$ will not be significantly different and as a result, it will be harder for the recognizer to identify the correct goal. A solution that address these constraints is proposed by E-Martin et al., which calculates the *interaction* of two or more actions (E-Martín, R-Moreno, & Smith, 2015) using the **plan graph** representation of the planning task (Blum & Furst, 1997). They first define the cost of two proposition/actions to be established together (*cost interaction*). This cost is propagated through the plan graph starting from the initial state until the goal states are achieved. The cost of achieving the goal is the sum of interactions between propositions and the costs of actions required to achieve that goal. Like Ramirez's solution, this work also assumes that the observations are not noisy but may be incomplete.

In certain problems, it maybe difficult to provide reasonable prior probabilities for the likely goals. This is specially true in the kinds of planning domains we are working on in this research, where certain facts about the domain are hidden to the actor and the absence of complete knowledge may enable some unintended goals, regardless of their priors. For example, consider a human user unwittingly falling victim to a phishing scam because he can not recognize phishing web sites from the safe ones.

Furthermore, in certain cases the recognizer may need to recognize the actor's plan before all the observations are made available. This is also an important requirement in the domains we are studying in this research, where an agent needs to assist human users avoid certain undesirable consequences before the human user's task is completed. This requires that recognition needs to happen as observations are made available incrementally. We will discuss some recent work in online recognition in the following sections.

Sohrabi et al. 2016 - Plan Recognition with Unreliable Observations

The recognition task is similar to the probabilistic plan recognition problem proposed by Ramirez and Geffner, where a recognizer receives an observation trace of the actor's agent's behavior and attempts to identify what the actor's plan is. They also use the "*Compiling Observations Away*" theory to generate a solution to the plan recognition problem. Sohrabi et al. propose several extensions to prior work where (1) the recognition system can now handle noisy or missing observations and (2) address observations over state variables. Further, the recognizer can identify the actor's plan and goals.

This work defines a **noisy** observation as an action that can not be explained by actions of a plan for a particular goal. **Missing** observation is an action that should have been in the observation trace but have not. When the observations are unreliable, the cost difference $C_g(O) - C_g(\overline{\mathrm{O}})$ will be large and as a result it will underestimate $P(g|O)$. This is because with noise in the trace, either the plan that is compatible with O will have a higher cost than the plan that is not compatible with O or the planner may not be able to find an observation compatible plan. To remedy this situation, they propose a new approach to "*Compiling Observations Away*" that modifies the planning domain to include action costs; specifically penalties for noisy/missing observations.

In Ramirez and Geffner's work, the observations are presented as actual actions as defined in the domain theory. Shorabi et al. argue that in realistic examples, it may be difficult for the recognizer to observe the action itself, but he may be able to observe the effect of the action in the state. Therefore, they define the observations over state variables instead of actions. When "*Compiling Observations Away*", they augment the domain D with special "discard" and "explain" actions. This classification helps to identify noisy observations that must be discarded. The definition for the plan recognition problem $\mathcal{R}$ closely follows from Ramirez and Geffner's definition, where $\mathcal{R} = \langle \mathcal{F}, \mathcal{A}, s_0, \mathcal{G}, O, PROB \rangle$. However, O is now a sequence of states, instead of actions. Recall that the domain theory $D = \langle \mathcal{F}, \mathcal{A}, s_0 \rangle$. The augmented domain theory $D' = \langle \mathcal{F}', \mathcal{A}', s_0' \rangle$. $\mathcal{F}'$ contains state variable predicates from D plus the new predicates $done, l_{o_0}$ to signal that the goal state is complete and to indicate the start of the O respectively. In addition $\mathcal{F}'$ has a set of special predicates l_{o_i} for each observation o_i indicating whether the observation was discarded or explained. The special predicates ensure the total order of observations is maintained when the new actions are executed. s_0' is equal to s_0 and has the additional predicate l_{o_0}. Goal $G' \in \mathcal{G}$ are expressed such that $G' = \{done, l_{o_m}\}$, where l_{o_m} is the last observation. There are four types of actions in $\mathcal{A}'$, each having costs associated with them **goal** action, **discard** action, **explain** action and original actions from $\mathcal{A}$. The *goal action* has preconditions equal to the goal state, postcondition equal to $done$ and cost = 0. The *discard action* is added when the observed state is false in the current state (i.e., observation not explained). As the post condition, the discard action adds the special predicate l_{o_i} and removes the special predicate added by previous observation $l_{o_{i-i}}$. Discard action cost = b_2 (a application specific weight). The *explain action* is added when the observed state is true in the current state. The post condition is similar to the discard action. The explain action has cost = 0. The *original action* has the same pre/post conditions from the original definition $\mathcal{A}$. However, it is now assigned a new cost, which adds a penalty proportionate to the number of missing observations. The plans derived from the augmented domain theory D', now reflect penalties to missing and unexplained observations.

The recognition task is performed by deriving posterior plan probabilities P($\pi|O$) and posterior goal probabilities P($G|O$) following the same process as Ramirez and Geffner.

- $P(\pi|O) = \beta P(O|\pi)P(\pi|G)P(G)$
- $P(G|O) = \beta P(O|G)P(G) = \beta \sum_{\pi \in \Pi} P(\pi|O)P(G)$

$P(G)$ is an input to the recognition problem as $PROB$. Note that, now $P(G|O)$ depends on $P(O|\pi)P(\pi|G)$. Sohrabi et al. approximates this value such that $P(O|\pi)P(\pi|G) \approx 1 - \frac{\beta V(\pi)}{\sum_{\pi' \in \Pi} V(\pi')}$, where $V(\pi)$ is a weighted cost factor that accounts for noisy and missing observations in a plan π for goal G that satisfies O. The term $\sum_{\pi' \in \Pi} V(\pi')$ is computed from the sampled plans that are derived from the augmented domain theory D' using Top-K planner (Riabov, Sohrabi, & Udrea, 2014), which finds k-best plans and using diverse plans. The recognized goal/plan is the goal/plan that has the highest posterior probability from the set of most likely goals/plans.

Constraints in the Approach

Similar to Ramirez and Geffner's work, this solution for plan and goal recognition relies on being able to provide the goal priors as an input to the algorithm. Furthermore, the observations (although given as a sequence of states) need to be provided to the algorithm up front. For the assistive agent we are designing for human users, the recognition process needs to be separated from these constraints.

2.4.3 Shvo et al. 2018 - Multi-agent Plan Recognition

So far, we have only discussed plan recognition in situations where there is one actor and one recognizer. An extension to this paradigm is proposed in work by Shvo et al. (2018), where the Multi-agent Plan Recognition problem (MAPR) is introduced. In MAPR the recognizer infers the goals and plans of multiple agents. The observations given to the recognizer originates from many agents. MAPR has interesting applications in multiple real domains such as intrusion detection, surveillance and so on.

When performing MAPR, the recognizer takes into account different capabilities of the agents in the domain. In addition, this work further considers actions taking some time to finish executing (durative actions). Similar to the work by Shorabi et al., observations are processed as states and not as actions. Furthermore, observations may be unreliable (i.e., contains missing/unexplainable observations). This solution to MAPR problem also builds on the plan recognition approach proposed by Ramirez and Geffner that proposes modifications to the domain theory to recognize goals/plans using automated planners. In the first step, the multi-agent aspect is compiled away into the domain theory. When there are many agents in the environment having different actions that they can execute, some actions can be executed concurrently, while others can not. They use temporal nature of actions to establish ordering constraints on the actions. Two special state predicates are defined for actions that operate as *action delimiters*: "start" and "end", which specify the pre and post conditions to the temporal action respectively. Additionally, they also define an overall precondition to the temporal action, which must hold true in every state between the "start" and "end" states. When actions are temporal, the solution plan is a sequence of action-time pairs showing actions that can execute at the same time when they are applicable. Use of temporal actions allows concurrency of the agent's own actions as well as actions of different agents. Now, in order to compile away the multi-agent aspect, they define a privacy model for the agents when executing the temporal (and perhaps shared) actions. Action a can be executed by agent i if and only if a is private to agent i or is public to all agents. This is done by introducing special predicates to track the ownership (i.e., which agent, what objects) of state variables when a temporal action is being executed. This privacy model is added to the initial state of the planning problem. The second step is "*Compiling Observations Away*". Here, they follow the same process as Sohrabi et al. 2016 to "*Compiling Observations Away*" for missing and noisy observations. Then, the plan recognition problem becomes finding the posterior probabilities of plans P($\pi|O$) and goals P($G|O$). Because the domain theory now has temporal actions, it is possible to use temporal planners to find sample solution plans.

Constraints in the Approach

The proposed approach is only evaluated for a case where the agents pursue one common goal. The assistive recognizer agent we are presenting in our research is an extension to this work. When different agents pursue different and competing goals, and when the recognizer is expected to assist one agent in the environment it needs to consider how likely the goal of the agent (who is being helped) will be threatened by competing agent(s) actions. The recognizer agent must be able to complete recognition *in time* to identify when a partial plan that seems to be helpful can be subverted by a competing agent during execution and alert the agent accordingly.

2.4.4 Planning as a Tool to Model User Behavior in Cyber-security

Cyber-security domain offers a lot of promise to study behavior both as normal users and as adversaries in automated planning. Behavioral Adversary Modeling System (BAMS) (Boddy, Gohde, Haigh, & Harp, 2005) uses automated planning to help computer network administrators in analyzing vulnerabilities in their system against various kinds of attacks. BAMS takes into account the properties of an adversary and produces plans that lead to system exploits that also coincides with the adversary model. While this work does not directly apply to plan recognition at its core, it illustrates a use case where classical planning can be used to design assistive systems targeted towards human end users.

This work models network vulnerabilities of a document management system as a planning problem and integrates a predictive behavior model of an adversary (who is a malicious insider) so that network administrators can concentrate on hardening the system in places where exploits are most likely and result of the exploit will be most costly. The network security planning domain contains objects such as email messages, files, hosts, user identifiers etc. and has 124 predicates to represents facts about the environment, status of tiles, capabilities, vulnerabilities of programs, knowledge possessed by users etc. Actions are represented in STRIPS with parameters, pre and post conditions. The domain has 56 actions that capture system events such as document accesses, user group modifications etc. The adversary's objectives are specified in the planning problem as goals. They use an off-the-shelf planner Metric-FF (Hoffmann, 2003) to find possible plans of the adversary. A graphical user interface allows end-users (i.e., network administrators) to configure problem specifications (hosts in the system, access control rules etc) and the attacker's properties (skills/tools he has) without having to encode them in PDDL.

The authors highlight several issues in modeling expressive scenarios in PDDL. The first constraint is the difference between the level of detail that is required in the domain and what can be modeled in the planning language. If the representation is too detailed, the resulting plans will be uninteresting and difficult for the users to extract information. Furthermore, developing and maintaining these highly descriptive domains will be a difficult task requiring expert knowledge. The second constraint is providing the plans generated by the automated planner to end-users in a natural representation. PDDL plans are not naturally comprehensible to end-users. The authors propose an encoding scheme for the PDDL actions and providing explanatory texts to capture state transitions that take place within the solution plan.

In this dissertation, we take a step toward designing assistive systems using automated planning to help human end users, who are non-experts (e.g., home users). Home users are specially vulnerable to undesirable consequences because they lack the know-how to recognize risky situations in advance. A previous study (Byrne et al., 2016) showed that home users pay more attention to the benefits of the activities than the risk; they have goals that they want/need to achieve and are willing to take the risk to achieve them. Many triggering actions may be normal activities (e.g., reading email, clicking on links) with the user more focused on the goal than on the risk. Thus, the undesirable consequence recognition problem needs to take into account the user's intention as well as the undesirable consequence.

Howe et al. (2012) observed that most studies that look into computer security practices of users relying on self reported surveys suffered from issues such as respondent bias, socially desirable responding and peer perception. The authors posited that experiments based on simulation, which place the participant in the actual situation that is monitored can help reduce such issues and also be leveraged to assess the emotional reactions of users to interventions and warnings.

The Intervention Problem can be directly applied in the cyber-security domain. An attacker attempting trick the user into compromising his security/privacy during day-to-day computing tasks fits the model of the competitor we discussed in this work. The attacker creates opportunities for phishing and malware attacks by making them to appear as common harmless tasks such as email and installing software. Unable to recognize these attacks in advance, the user becomes an unwitting accomplice to security breaches.

2.5 Goal Recognition

Similar to plan recognition, early goal recognition solutions involved using plan libraries. Lesh and Etzioni (1996) use a plan library based approach for goal recognition. Here, they use a data structure called a *consistency graph*, which is a directed graph with actions, action schema, and goals as nodes. There is an edge between two nodes if there is a *consistent* plan that supports the two nodes for some goal. Their approach to recognizing an actor's goal rely on quickly determining if a goal is inconsistent with the observed actions. Informally, the recognizer needs to conclude that the actor could not possibly have executed the observed actions as a part of a plan to satisfy the goal. This requires the recognizer to reason about **all** plans for each candidate goal. Consistency graph is an efficient way to represent planning problems and reason about them tractably.

Similar to the plan recognition problem, the goal recognizer takes as input a set of likely goals, a sequence of actions executed by the actor, actor's beliefs and a model of action schema that can be executed in the domain. The solution to the goal recognition problem is a subset of the goal schema such that, for every element in the subset, there exists a plan that achieves that goal. In order to find this goal schema subset, they first build the consistency graph. Then they gradually remove elements (action schemas/goal schemas) following a predefined rule set, which recognizes elements in the graph that are not in any consistent plan without violating the graph correctness.

Actions are removed from the consistency graph when the observed action is not consistent with other actions in the plan or the goal. This rule is sensitive to noisy observations. Oftentimes, human users do not take actions to achieve a goal in a rigid sequence. For example, they may try a few options before settling into one plan or they could just be *exploring* the domain. This is specially true in the kinds of domains we are investigating in this research: cyber-security and Rush Hour. Actions generated in these type of situations may eventually be removed from the graph, which results in the recognizer failing to identify the actor's goal.

2.5.1 Ramirez et al. 2009 - Plan Recognition as Planning

This work is the precursor to the probabilistic plan recognition work proposed by Ramirez and Geffner that we discussed in plan recognition related work. There are many similarities between plan recognition as planning (Ramírez & Geffner, 2009) and probabilistic plan recognition. As stated in the previous work by the same researchers, the environment E and the recognizer's problem are defined the same. They both use the domain theory to generate two types of plans to find the most likely goal: (1) one that is compatible with observations and (2) one that is not compatible with observations. The key differences are the assumption that the actor is only executing optimal plans and the approach they use to compile the observations away. The "*Compiling Observations Away*" technique used in this work to modify the original domain theory D into a new domain theory D' by adding new action definitions and predicates corresponding to the observations.

Let us define $D = \{\mathcal{F}, \mathcal{A}, s_0\}$ for plan recognition problem illustrated in Figure 2.3. $\mathcal{F} = \{(at\ x), (adj\ x,\ y), L1\_1, L1\_2, \ldots, L8\_5\}$, which indicates that the actor is at location x and location x is adjacent to location y respectively. Predicates $L1\_1$ etc. refer to the cells on the grid, corresponding to the row and the column numbers of the cell. $\mathcal{A} = \{move(x, y)\}$, which indicates that the agent can move from location x to location y. Preconditions of the move operation $pre(move) = \{(at x) \wedge (adj\, x, y)\}$. Post-conditions for the move operation $add(move) = \{(at y)\}$, $del(move) = \{(at\ x)\}$. The initial state $s_0 = \{(at\ L5\_3)\}$. Let us assume the recognizer gets the observation trace $O = \{move(L5\_3, L4\_3)\}$, which indicates that the actor has moved up one cell from the initial position.

If we are to apply the *Compiling Observations Away* theory proposed by Ramirez and Geffner to transform D to $D' = \{\mathcal{F}', \mathcal{A}', s_0'\}$ adds one new state variable predicate such that $\mathcal{F}' =$

$\mathcal{F} \cup \{p_{move(L5\_3,L4\_3)}\}$ and one new action such that $\mathcal{A}' = \mathcal{A} \cup \{ob_{move(L5\_3,L4\_3)}\}$. The new state variable predicate indicates that the an observation has occurred in the environment (in this case, move up). The new action is fully grounded and defined below.

operator: $\mathtt{ob}_{\mathtt{move}(L5\_3,L4\_3)}$

preconditions: $\{$ $(\mathtt{at}\ L5\_3) \wedge (\mathtt{adj}\ L5\_3, L4\_3)$ $\}$

postconditions: $\{$ $\neg\, (\mathtt{at}\ L5\_3) \wedge (\mathtt{at}\ L4\_3) \wedge \mathtt{p}_{\mathtt{move}(L5\_3,L4\_3)}$ $\}$

Comparing this to the compilation proposed in probabilistic plan recognition, the probabilistic method does not add any new action definitions to the domain theory. Instead, only the new state variable predicates are added into existing action definitions. In order to force the planner (an optimal planner in this case) to find solutions that are optimal in number of moves and also contains actions in the observations, the goals $\mathcal{G}$ are also modified to reflect the effects of the observation. For example, the goal $g \in \mathcal{G}$ such that $g = \{(\mathtt{at}\ L1\_2)\}$ is modified to $g' = \{(\mathtt{at}\ L1\_2) \wedge \mathtt{p}_{\mathtt{move}(L5\_3,L4\_3)}\}$. The process of adding new state variable predicates and new actions is repeated for all actions in the observation trace. All goals in the candidate goal set are also modified accordingly. The likely goal given observations ($P(g|O)$) is found by the same calculation as the probabilistic plan recognition by taking the cost difference between optimal plans that contain the observations and optimal plans that do not contain the observations.

Constraints in the Approach

This work also has the same constraints as the probabilistic plan recognition, in that the goals are recognized when all the observations are available and the difficulty in specifying the goal priors. Plan intervention requires us to identify undesirable consequences before the task is completed. It follows that the offline model of recognition is not suitable for plan intervention. However, we will use the plan recognition with "*Compiling Observations Away*" theory to benchmark our initial plan intervention solutions.

2.5.2 Vered et al. 2017 - Online Goal Recognition in Continuous Domains

So far, the goal and plan recognition problems we have discussed in this chapter assume the actor and the recognizer are in a discrete domain and the observations are discrete. Online goal recognition (Vered & Kaminka, 2017) extends the recognition problem to continuous domains. For example, in robot motion planning, we can define a simple domain as the space of possible positions for a robot. This definition can further be extended to define higher order dimensions such as angle, velocity etc. Further, the recognition is online, which means that the recognition problem must be solved for every new observation when they are revealed. The recognizer receives observations of the actor's position in the same n-dimensional space (e.g., a point or a trajectory). The recognition problem them becomes finding the goal g in the candidate set of goals $\mathcal{G}$ that best matches the observations. Formally, we seek to determine $P(g|O)$ for each goal $g \in \mathcal{G}$. The recognized goal is the one that has the highest posterior probability.

They propose a new method for ranking goals in $\mathcal{G}$. Instead of taking the cost difference (Ramirez and Geffners' approach) they define a ratio $score(g) = \frac{cost(i_g)}{cost(m_g)}$, where i_g is the *optimal* plan to achieve g and m_g is the *optimal* plan that achieves g and includes all the observations. When the optimal plan that has all the observations is the same cost as the optimal the score approaches 1. Then $P(g|O) = \eta score(g)$, where η is the normalizing constant. i_g can be computed using a planner. To compute m_g, they exploit the fact that each observation is a trajectory or point in the continuous space and each likely plan is also a trajectory in the same space. Therefore $m_g = prefix + suffix$, where $prefix$ is built by concatenating all observations in O into a single trajectory, and the $suffix$ is generated by calling a planner from the last observed point to goal g. To improve the computation efficiency during recognition, they introduce two functions: RECOMPUTE - recomputes the new plans only if the new observations seem to change the plan significantly, and PRUNE - removes unlikely goals from $\mathcal{G}$.

Constraints in the Approach

Although the recognition algorithm is evaluated in continuous domains, follow up work extends this solution to discrete domains (Vered, Pereira, Magnaguagno, Meneguzzi, & Kaminka, 2018).

The planning applications we are studying in this research are modeled for discrete domains. We will use goal ranking heuristic proposed in this solution to benchmark our initial plan intervention solutions.

The proposed online goal recognition approach further reduces the computational cost by introducing *landmarks* to prune the likely goals (Vered et al., 2018). Landmarks are facts/actions that must be true/executed at some pin all valid plans that achieve a goal from an initial state (Hoffmann et al., 2004). This work uses landmarks to heuristically estimate the goal completion ratio (i.e., more landmarks are active in the current state means that particular goal is closer to being achieved) as a proxy for estimating $P(g|O)$. Landmark based heuristics are often times used to improve run time during the goal recognition process. Pereira et al. use a landmark based heuristic to estimate the proximity to each goal (Pereira, Oren, & Meneguzzi, 2017). This heuristic measures the ratio between achieved and non-achieved landmarks.

2.6 Recognition and Interaction

In the goal/plan recognition work discussed so far, the recognizer passively observes the actor. The recognizer's task finishes once the actor's goal/plan is identified. An extension to this model is to allow the recognizer to interact with the environment and affect the actor's behavior. This interaction can be **offline**: where the recognizer modifies the domain before the actor can execute any plans on it, **online**: provoke the actor to behave in some specific way by setting values of environment features, and **direct communication**: where the actor is asked questions such that the goal hypotheses can be pruned quickly.

When designing assistive intervention models for human users, in addition to recognizing what they are trying to achieve in the domain, we must provide some guidance if it can be determined that the goal they are trying to accomplish has become unachievable. Prior work in goal/plan recognition provide some insight on how this can be achieved by allowing the recognizer to interact with the actor. We discuss some approaches below.

2.6.1 Keren et al. 2014 - Goal Recognition Design

Goal recognition design (GRD) (Keren, Gal, & Karpas, 2014) proposes a solution that allows the observer to measure the "difficulty" of performing goal recognition in the current domain and propose modifications (e.g. blocking actions) such that goal recognition can be done easily. GRD is an offline interaction activity, which means that unlike the typical plan/goal recognition that relies on having an observation trace for the actor's behavior, GRD analyzes the plans for possible goals in the domain an actor might execute and evaluates how distinct the plans are.

This work introduces a new metric *worst-case distinctiveness* (*wcd*) that measures the maximum length of the common prefix all plans for the likely set of goals may share in the current design of the domain. The solution to the GRD problem is a modified domain that assures *wcd* is minimized, when plans are generated optimally. It also assumes that the actions are deterministic and fully observable.

2.6.2 Pozanco et al. 2018 - Counterplanning using Goal Recognition and Landmarks

In multi-agent settings the agents in the domain may be adversarial. This means that the agent may want to prevent another agent from achieving its goal. This work presents a domain independent approach for counterplanning based on goal recognition, landmarks and automated planning. The design of the counterplanning domain is such that there are two adversarial agents (seeking agent and preventing agent). The two agents pursue different goals. The recognizer's task is to help the preventing agent block the seeking agent from reaching his goal.

Counterplanning requires that the recognizer quickly identify the seeking agent's goal. This work uses Ramirez and Geffener's probabilistic goal recognition algorithm to perform goal recognition. Next, the recognizer needs to identify the earliest landmark for the seeking agent's planning problem (for the recognized goal) that needs to be blocked (i.e., counterplanning landmark). A counterplanning landmark is given a set of fact landmarks from the seeking agent's planning task, the counter planning landmark is a postcondition of an action the preventive agent can execute. If the counterplanning landmark is a positive fluent, the preventive agent's action must delete it. If the counterplanning landmark is a negative fluent, the preventive agent's action must add it. The recognizer's interaction occurs when he uses automated planning to generate a plan to achieve the counterplanning landmark (e.g., negating the landmark), and therefore blocking the seeking agent's goal achievement.

2.6.3 Mirsky et al. 2016 - Sequential Plan Recognition

In goal recognition, the recognizer reasons about how likely a given set of possible goals are given the actor's behavior. The recognizer's task may fail if he can not accurately disambiguate between possible goal hypotheses. This work proposes a solution that involves the recognizer querying the actor about whether a candidate plan in one of the goal hypotheses matches the actor's intention. During the recognition process, the actor is sequentially queried in real-time whether the observed partial plan is correct. The actor's answers are used to prune the possible hypotheses, while accounting for the incomplete plans that could match with the observations after several other observations happen in the future. In order to optimize the querying process, the recognizer considers only the queries that maximizes the information-gain and the likelihood of the resulting hypotheses given the expected query result.

This solution assumes that a plan library is provided to the recognizer in advance. Their implementation of the plan library uses trees to represent the possible plans for goal hypotheses. The planning domain used in this study describes how to perform chemistry lab experiments using an educational software that simulated a virtual lab. The plans in the planning library were traces were taken from real students' traces when interacting with the virtual chemistry lab.

2.6.4 Goal/Plan Recognition with an Active Observer

In real-life situations, helpful intervention requires more observer engagement, i.e., an active observer to help the actor recover from intervention. Therefore, we discuss existing work on designing active observers having both the recognition and interaction capabilities. Next, we discuss related work on dealing with misconceptions held by the actor about the planning domain and AI safety in general. Because we specially focus on developing intervention models for human users, we discuss related work on using machine learning algorithm to classify human user behavior and how intervention is leveraged to provide intelligent help to human users.

While the goal/plan recognition works discussed in the previous section assume a passive observer, a growing body of work has also looked into recognition problems with active observers. Only recognizing when intervention is needed (as a passive observer) solves only a part of the problem. In cases where intervention is used for an artificial agent, active observers can force the agent to alter its current plan. When intervention happens during a cognitively engaging task, as in the Rush Hour puzzle, a human user would naturally like to know what to do next. An active observer who can take action or give instructions to the human user, not only will be able to assist the user complete the task safely but also will improve the human user's interaction with the AI system.

(2011) propose a plan library based plan recognition technique to provoke the observed agent so that it becomes easier to disambiguate between pending goal hypotheses. The observer modifies the fluents associated with a *provokable* action, which forces the observed agent to react on the modification. The provokable event is selected heuristically such that it reduces the uncertainty among the observed agent's likely goals. In another approach that aims to expedite the goal recognition, (2020) use landmarks to eliminate hypothesized goals. They define the Active Goal Recognition problem for an observer agent who can execute sensing and world-altering actions. The observer executes a *contingent plan* containing the sensing and world-altering actions to confirm/refute the landmarks of the planning problems for each goal hypothesis. Goals hypotheses whose landmarks (for the corresponding planning problem) are refuted by the execution of the contingent plan are removed from the set of likely goals. Although the initial problem definition assumes that the observer's contingent plan is non-intervening and is primarily used to reduce the goal hypotheses, (2020) also propose an extension where the observer can actively impede or aid the actor. For example, the authors suggest adopting the Counter-planning Algorithm proposed by (2018) to generate a plan for the observer to impede the actor, after the actor's goals are identified through Active Goal Recognition. Pozanco's Counter-planning Algorithm is designed for a domain where two adversarial agents (seeking and preventing) pursue different goals. In the context of the Active Goal Recognition problem, the seeking agent is the actor while the preventing agent is the observer. Counter-planning requires that the observer quickly identify the seeking agent's goal. They use the Ramirez and Geffener's probabilistic goal recognition algorithm to perform goal recognition. Then the preventing agent actively intervenes the seeking agent by identifying the earliest landmark for the seeking agent's planning problem (for the recognized goal) that needs to be blocked (i.e., counter-planning landmark). The recognizer uses automated planning to generate a plan to achieve the counter-planning landmark (e.g., negating the landmark), thus blocking the seeking agent's goal achievement. The aforementioned works in Active Goal Recognition assume full observability over the actor. (2019) relax this constraint and propose Active Goal Recognition with partial observability over the actor and model the planning problem as a partially observable Markov decision process (POMDP) (Kaelbling, Littman, & Cassandra, 1998). Similar to the previously discussed Active Goal Recognition problems, the observer agent is trying to reach it's own goal as well as correctly predict the chosen goal of the actor. Therefore, they define the Active Goal Recognition problem for the observer by augmenting the observer's action space with the actor's actions, the observer's own actions and the decision actions on the actor's goals. The state space is defined as the Cartesian product of the observer's states, actor's states and actor's goals. The goals for the recognition problem are augmented with the observer's own goals. and the prediction of the actor's goals. A solution to this planning problem starts at the the initial states of the observer and the actor and chooses actions to the augmented goal while minimizing the cost (or maximizing a reward). A POMDP is defined to solve the augmented planning problem (i.e., Active Goal Recognition problem).

The goal recognition algorithms discussed above mainly focus on pruning the pending goal hypotheses to allow the observer quickly disambiguate between goals. To accomplish this objective, Shvo et al. use sensing and world-altering actions to confirm/refute the landmarks. Counter-planning also uses landmarks. Bisson et al. use heuristics. Other solutions for goal recognition take a decision theoretic approach where the observer attempts to find plans to achieve own goals while predicting the user's goal optimizing over some reward function. Our intervention models differ from these solutions in the intervention recognition task because we do not prune the goal hypotheses. Instead, we emphasize on accurately recognizing whether an actor's revealed plan is unhelpful (and must be interrupted) where the plans leading to the goal hypotheses share common prefixes, making the disambiguation difficult. We use machine learning to learn the differences between the helpful and unhelpful plan suffixes and use that information to decide when to intervene. We rely on the same plan properties as existing recognition algorithms to learn the differences between plan suffixes: plan cost and landmarks. In addition, we have shown that the plan distance metrics can also be used to differentiate between helpful and unhelpful plan suffixes.

The next step in our work is to extend the Human-aware Intervention model so that the observer can actively help the human user modify his plan following the recognition phase. The works we discussed in this section have already addressed this requirement in agent environments, where the observer also executes actions to support the goal recognition process. Pozanco et al. take a step further to show that following recognition, the observer can impede the actor using planning. (2017) discuss a method that allows the observer to interact with the actor while the actor's plan is in progress with fewer observations available. Our experiments validate their argument that plan/goal recognition by itself is more useful as a post-processing step when the final actions are observed, which will be too late for the Intervention problems we discuss in this work. Our solution addresses this limitation, allowing the observer to recognize "before it's too late" that the undesirable state is developing. We use machine learning to perform the recognition task. In contrast, (2017) propose a domain modification technique (similar to Ramirez and Geffner's) to formulate a planning problem that determines a relevant interactive response from the current state. Plans that agree (and do not agree) with the observations can now be found using an off-the-shelf planner on the modified domain. The actor's goal is recognized by comparing the costs of these plan sets. Following the recognition phase, they also define assistive and adversarial responsive actions the observer can execute during the interaction phase. Assistive responsive action generation is more related to our

Intervention problem because our observer's goal is to help the actor avoid the undesirable state. The authors define an assistive interaction planning problem to generate a plan from the current state for the observer. This *assistive* plan uses the combined fluents of the actor and the observer, the Cartesian product of the actor's and the observer's actions (including no-op actions) and a modified goal condition for the observer. The assistive action generation through planning proposed by (2017) is a complementary approach for the interactive Human-aware Intervention model we hope to implement in the next phase of this work. However, we will specially focus on using automated planning to inform the decision making process of the human actors following intervention. In addition, our work in Unsafe Suffix Recognition can be further extended by relaxing the assumptions we have made in the current implementation about the agents and the environment, specifically deterministic actions and full observability for the observer. This may require adopting planning techniques like the one proposed by Amato et al., but with different reward functions. For example, for intervention problems the reward function may take into account the freedom of the actor to reach his goal while ensuring safety.

Dealing with Misconceptions Held by the Actor

In our intervention model the undesirable state is hidden to the user. This is similar to the user having a misconception or a false belief about the domain as the user "believes" the undesirable state is actually safe. Although for our Intervention problem, we assume that the user's belief model is explicitly available to the observer, in other situations this assumption may not hold (e.g., the observer may have limited sensing capabilities). In this case, another agent in the environment (like the observer in our Intervention problem), needs to be able to acquire the beliefs the user has. (2014) discuss a belief acquisition process for a search and rescue domain. The belief acquirer maps the beliefs into a planning problem, allowing him to predict the plan of the agent who is missing the beliefs. The predicted plan and the belief acquirer's own plans are then used to achieve coordination among human-robot teams.

(2020), in Epistemic Multi-agent Planning, use a multi-agent modal logic to model an observer (and other actors) having different beliefs about the world and other actors. This is in contrast to (2014), who use First-order logic. Given an Epistemic Plan Recognition problem (for an Epistemic Planning observer and an actor), the authors define an *ill-formed* plan with respect to some goal if and only if the plan achieves the goal with respect to the actor but does not achieve the goal from the observer's perspective. The authors highlight a limitation of Epistemic Plan Recognition (also applicable in normal Plan Recognition). The observer's recognition efficacy is dependent on the completeness and the veracity of the observer's beliefs about the environment and the actor. In addition it is also limited by how distinguishable the goals and the plans are that need to be recognized. Our intervention solution attempts to address the problem of improving recognition accuracy when plans are indistinguishable. (2020) introduce *adequacy* for the recognition process when the actor's actual beliefs are different from the observer's beliefs about the actor. If the observer's beliefs about the actor's beliefs are *adequate*, then the observer can generate precisely all plans that the actor can also generates for some goal that also satisfies the observations.

2.6.5 AI Safety

Using our Intervention models an observer can recognize, with few false alarms/misses, that an undesirable state is developing. The recognition enables the observer to take some action to help the user avoid the undesirable state and complete the task safely. Therefore, our work is also a precursor to incorporating safety into AI systems.

(2018) use factored Markov Decision Process to model a domain where an agent, while executing plans to achieve the goals that are desirable to a human user, also wants to avoid the negative side effects that the human user would find undesirable/surprising. The agent has complete knowledge about the MDP, but does not know about the domain features that the user has given permission to change. In order to find the safety optimal policies, the agent partitions the domain features as *free*, *locked* and *unknown* (treated as locked). Then the MDP is solved using linear programming with constraints that prevent the policy from visiting states with changed values for the locked, unknown features. The feature partitioning is similar to our analysis of safe and unsafe plan suffixes using features of plans, where we explore the plan space to recognize what plans enable/satisfy the undesirable state and what do not. In contrast to their model, we model the agents' environment as a deterministic domain using STRIPS. (2018) policy generation process interacts with the user (through querying) to find the safe-optimal policies that the user really cares about. (2020) propose a multi-objective approach to mitigating the negative side-effects. Given an task modeled as a MDP, the agent must optimize over the reward for the assigned task (akin to the desirable goal in our Intervention problem), minimize the negative side effects (the undesirable state) within a maximum expected loss of the reward for the assigned task (*slack*) in order to minimize the negative side effect. Being able to handle the negative side effects, caused by imperfect information in the environment is also pertinent to Human-aware Intervention that we propose. Although in this work we are more focused on intervention recognition than intervention response, it's also important to consider how the user's feedback/preferences can be factored into intervention recovery for more robust human-agent interaction.

(2016) introduce *cooperative inverse reinforcement learning* (CIRL) to ensure that the autonomous system poses no risks to the human user and align it's values to that of the human in the environment. The key idea is that the observer (a robot) is interactively attempting to maximize the human's reward while observing the actions executed by the human. The cooperative game environment is modeled as a Partially Observable Markov Decision Process and the reward function incentivizes the human to teach and the robot to learn, leading to a cooperative learning behavior. The problem of finding the optimal policy pair for the robot and the human is found by reducing the problem to solving a partially observable Markov decision process. Intervention is a continuous process where the user and the agent will interact with each other repeatedly until the task is complete Especially in helpful intervention (like the idea we propose in this work, repeated interaction allows the human user and the agent to learn more about the task and hopefully complete it safe-optimally. CIRL formalizes a solution to address this problem.

2.7 Human User Behavior Classification

The design of assistive agents for human users require that the recognizer be able to identify human behavior and how well the behavior aligns with the goals of the system they interacts with. Human users are not always rational and may have hidden goals. It maybe an unfair comparison to model them as rational agents in real life scenarios.

Behavior classification is different from typical plan/goal recognition. It aims to achieve some insight about the actor from the passive observer's perspective. The work proposed by Borrajo et al. (2020) discusses the design of an observer agent, which tries to learn characteristics of other agents by observing their behavior when executing actions in a given environment. Using the financial transactions domain as a case study, this work models two agents: the actor (e.g., a bank customer) and an observer (e.g., the banking institution). Only the actor can execute plans in the environment. The observer does not know the actor's goal and has partial observability of the actor's behavior (actions the actor executes). Then, the observer's task is to classify the observed behavior into different types of known behavior classes. In order for the application to be domain-independent, the authors use plan distance measures (e.g., Jaccard similarity) between observed actions and distance between observed states as features to train the classifier. Given two plans p and p', the Jaccard similarity for the actions in the plan is defined as: $\frac{|A(p) \cap A(p')|}{|A(p) \cup A(p')|}$, where $A(p)$ and $A(p')$ refer to the sets of actions in the plans p and p' respectively. We use a similar features to recognize the actor's plan prefixes that lead to undesirable states.

2.8 Providing Intelligent Help to Human Users Through Intervention

(2002a, 2002b) discuss the design of a system that provides intelligent help for novice human users while using a file manipulating software application. The Intelligent File Manipulator (IFM) is an online help system where it automatically recognizes that an action may not have the desired goal for the user and offers help by generating alternative actions that would achieve the user's goals. IFM uses a user modeling component to reason over the observed actions. The user modeling component combines a limited goal recognition mechanism and a simulator for users' reasoning based on Human Plausible Reasoning theory to generate hypotheses about possible errors the user might make.

There are some similarities between IFM and our proposed intervention framework. Both models use observations of actions as input for deciding intervention. Both models assume that the user's goals are known. We now discuss some differences between the IFM intervention model and our proposed model: Intervention by Suffix Analysis. The IFM domain is modeled as a task hierarchy, while our domain models (benchmark and Rush Hour) are sequential. To map the user's observed sequence of actions to the plans leading to the desirable and undesirable states, our intervention model uses automated planning to explore the plan space. Then, we analyze the remaining plan suffixes using machine learning to decide intervention. IFM does not use automated planning. Instead it uses a limited goal recognition mechanism called "instability" to identify when users need help. They identify a set of states of the file system as undesirable such as empty directories, multiple copies of a certain file etc. If the file system state contains any of the preset undesirable states, then the system contains instabilities. The user's action will either add an instability or remove an existing one from the system's state. The system tracks the progress of the user's plan(s) by monitoring how the instabilities are added and removed from the system. IFM categorizes the user's observed actions into four categories "expected", "neutral", "suspect" and "erroneous" depending on how compatible the observed actions are with the user's hypothesized intentions. Intervention in IFM takes place when the user executes "suspect" or "erroneous" actions because they signal that there are still unfinished plans. To help the user recover from intervention, the IFM flags "suspect" or "erroneous" actions, and suggests alternative actions that are compatible with the user's intentions. Finding the alternative actions similar to the ones the user has already executed is done based on the user models derived from the Human Plausible Reasoning theory. We hope to address the issue of intervention recovery for our proposed Human-aware Intervention Problem in future developments of our application.

(2016) present HEALER, a software agent that sequentially select persons for intervention camps from a dynamic, uncertain network of participants such that the spread of HIV/AIDS awareness is maximized. Real-life information about the nodes of the network (human users) are captured and modeled as a POMDP. The Intervention problem discussed in this work is slightly different from our model. Solving the POMDP gives the solution for how to select the most influential individuals from the network to maximize awareness among the population. In contrast, our intervention model is defined for a discrete and sequential environment. A similarity between the models is that they codify properties of actual human users into the POMDP so that the model can be adopted in real-life application. Our Human-aware Intervention model too is designed from actual human user data.

A body of literature on managing task interruption focuses on using cognitive modeling to predict human behavior, which can be used to identify intervention points. Hiatt et al. ((2011)) apply theory of mind to accommodate variability in human behavior during task completion. They show that a theory of mind approach can help explain possible reasons behind a human's unexpected action, that then allows the robot to respond appropriately. Ratwani et al. ((2008)) demonstrate that a cognitive model can accurately predict situations where a human missed a step in a sequence of tasks. More recently, Altmann & Trafton ((2020)) show how to extend a cognitive model to explain a cognitively plausible mechanism for tracking multiple, interacting goals.

2.8.1 Intervention in Cyber-security for Home Users

Cyber-security domain offers a lot of promise to study behavior both as normal users and as adversaries in automated planning. Behavioral Adversary Modeling System (BAMS) (Boddy et al., 2005) uses automated planning to help computer network administrators in analyzing vulnerabilities in their system against various kinds of attacks. BAMS takes into account the properties of an adversary and produces plans that lead to system exploits that also coincides with the adversary model. While this work does not directly apply to plan recognition at its core, it illustrates a use case where classical planning can be used to design assistive systems targeted towards human end users.

In this work, we take a step toward designing assistive systems to help human end users, who are non-experts (e.g., home users). Home users are specially vulnerable to undesirable consequences because they lack the know-how to recognize risky situations in advance. A previous study (Byrne et al., 2016) showed that home users pay more attention to the benefits of the activities than the risk; they have goals that they want/need to achieve and are willing to take the risk to achieve them. Many triggering actions may be normal activities (e.g., reading email, clicking on links) with the user more focused on the goal than on the risk. Thus, the undesirable consequence recognition problem needs to take into account the user's intention as well as the undesirable consequence.

(2012) observed that most studies that look into computer security practices of users relying on self reported surveys suffered from issues such as respondent bias, socially desirable responding and peer perception. The authors posited that experiments based on simulation, which place the participant in the actual situation that is monitored can help reduce such issues and also be leveraged to assess the emotional reactions of users to interventions and warnings.

The Intervention Problem can be directly applied in the cyber-security domain. An attacker attempting to trick the user into compromising his security/privacy during day-to-day computing tasks (e.g., reading email, installing software) fits the intervention model with the user, competitor and the observer we discussed in this work. Given a cyber-security planning domain model with sufficient complexity (e.g., BAMS domain model), where the undesirable state (i.e., security breach) may develop over time, the Unsafe Suffix Analysis Intervention model can be applied to recognize the threat in advance. A key requirement in helping users in cyber-security domain is to minimize the false positives and negatives during intervention recognition. As evidenced by the experiment results on benchmark domains and the Rush Hour domain confirm, our proposed learning based algorithm addresses this requirement well. While our approach uses Automated Planning, a complementary approach proposed by (2011) use Attack Graphs to model vulnerabilities in an intrusion detection system to detect attack scenarios while decreasing false positives. However, the intervention recognition must also be paired with intervention recovery in cyber-security domains to ensure the safety of the agent or the human user, particularly when the user has partial visibility or limited capability for understanding the severity of threats. Intervention recovery is also important in help the agent or the human user safely complete the task.

Chapter 3

How Do Home Computer Users Behave in Questionable Security Situations?

In this chapter, we present the findings from a human subject experiment that studied human behavior in practicing computer security. In this study, we simulate cyber-security vulnerabilities that occur in a home computer environment when the human users perform routine tasks like checking email and using software applications. In our analysis, we find that the human users' intentions and post-hoc perceptions of computer security generally are not indicative of their actions; they were unaware when they had triggered security or privacy breaches. The findings of this study motivated our work on plan intervention. In the next chapters we discuss three plan intervention models that can automatically detect an undesirable state developing and decide when to intervene. Furthermore, our findings of this study support the claim that security solutions designed for end users to practice safe computing need to assess two information sources:

- The human user's decision making context
- The factors that affect the user's decision making ability
- Observational data like system logs and user action sequences

3.1 Introduction

The 2016 American Community Survey found that 89% of all households had a computer, including smartphones, making it a common feature of everyday life (Ryan, 2017). Also, 81% had a broadband subscription. The Internet impacts many areas of the daily life; from performing routine tasks like shopping, banking and connecting with family and friends. The Internet has become an avenue for pursuing formal education (e.g., online degree programs) as well as informal learning (e.g., how-to videos on cooking, household repairs etc.) and allows us to collaborate across many physical barriers. Whether we realize it or not, we are routinely deciding whether or not to take risks in security and privacy when we interact with computer systems online.

Home computer use, as distinguished from work related or organizationally based computer use, is dominated by personal activities (mostly recreational, but some financial and health related) and is not governed by security policies determined by experts. Most security approaches for home users encourage them to take preventative or precautionary actions, such as installing anti-virus/malware software, hardening passwords (Chiasson, Forget, Biddle, & van Oorschot, 2008), backing up information (Dupuis, Crossler, & Endicott-Popovsky, 2012). In this study, our objective is to understand how home computer users make more *immediate* security and privacy decisions, such as whether to click on links in email messages, to install software or visit suspect websites.

We distinguish two decision making contexts for home computer users: unwittingly taking action that may lead to undesirable consequences (e.g., clicking on links in phishing emails) and deciding to accept the possibility of undesirable consequences (e.g., financial transactions on trusted sites or P2P software installations). In the first case, the user does not recognize that the situation may have undesirable consequences. Thus, our research in the "**unwitting context**" will be to initially assist the user in better identifying possible undesirable consequences of action. We will do this by designing methods for automatically recognizing security/privacy risks. We discuss intervention solutions that address this requirement in Chapters 4 and 5. In contexts where the user recognizes the consequence, the research will address how to help users frame their decisions: determining what alternatives are available for the user to take as next steps. We discuss an intervention solution that address this requirement in Chapter 6 of this dissertation.

While computer security and privacy decision making share characteristics of other decision making models about possibility of undesirable consequences, the combination of decision making models within the context of practicing cyber-security is unique. Thus, theoretical models (e.g., Protection Motivation Theory (Rogers & Prentice-Dunn, 1997) and Theory of Planned Behavior (Ajzen, 1991)) applied in other decision-making contexts (e.g., health) fall short in their applicability because they use different models of risk management. Because of the diversity of the population and decision contexts, we favor a personalizable approach in which computer assistance can be tailored to the individual user and the decision making context.

The study discussed in this chapter makes several contributions:

- Analyses and human subject studies relating what human users think is happening to what they actually do. This is critical to understanding how much information about user's decision making context can be inferred from surveys.

- Studies and analyses relating perceptions, actions and some user characteristics to possible interventions.

- Software framework to support home computer user security/privacy studies. We create a lightweight, sandbox software system, which simulates the Microsoft Windows environment on desktops and laptops. The simulator supports a small set of vulnerabilities and the scenarios that could possibly trigger them.

The decision of when to flag a problem for intervention also relies on the knowledge about the user. We refer to this as the ***user's decision making context***. Success of the intervention model will depend on not bothering the user unnecessarily.

For example, consider cyber-security threats like phishing attacks and spoofing attacks. If users have the skill for recognizing these cyber-security threats and understand the consequences, then they need less monitoring. In contrast, users who are less aware of the computer security threats require more help. So, the question we need to address is *what type of users make particular computer security decisions*. We answer this question by drawing from health related behavioral models to explain security behavior. Health behavior is a natural analogy to how human users respond to cyber-security threats. Human users frame many related issues in cyber-security using health related metaphors. For example, we refer to viruses and infections when talking about cyber attacks, we discuss system hardening after an attack takes place.

The Health Belief Model (HBM) is focused on user's attitudes and beliefs. The user's beliefs are described based on their perceptions of six factors, susceptibility, severity, benefits, barriers, cues-to-action, and self-efficacy of performing a given health behavior (Rosenstock, Strecher, & Becker, 1988). In this work, we study the influence of two user characteristics when making security decisions from the HBM: *self-efficacy*, that is an individuals confidence in her/his ability to perform a security enabling task on the computer, and *cues-to-action*, that is an individual's response to external triggers for how they would affect users practicing computer security. We are adopting the HBM for studying user characteristics that affect computer security decision making because the HBM (specifically self-efficacy and cues-to-action) has been studied in prior work, which looked into computer security practices of the home user. Studies on self efficacy found that (Urbanska, Roberts, Ray, Howe, & Byrne, 2013; Aytes & Connolly, 2004; Milne, Labrecque, & Cromer, 2009) knowledge about how to practice safe computing increases the rate of safe behavior. However, having knowledge alone does not guarantee practice of computer security. For example, rapid advances in new security technologies make it harder for a user to keep up with the latest updates that must be made to ensure safety of his computer. Cues-to-action triggers are advice received from external entities like friends, co-workers, media reports, laws and regulations etc. Studies on cues-to-action (Claar, 2010; Ng & Xu, 2007) find that these types of cues do not have a significant effect on human users practicing computer security. However, these studies mainly use self-reports to evaluate the effects of self-efficacy and cues-to-action on users' decision making. In this work, our objective is to find out whether the same effects can be observed from human users practicing computer security in a simulated home computer environment. Furthermore, the simulated environment allows us to present cues to users that can be monitored easily such as HTTPS/HTTP web sites and safe/unsafe hyperlinks. This enables us to measure user responses to measurable cues as opposed to cues obtained from external sources, which are often hard to monitor and control.

Considerable research has addressed human decision making about risk. An individual's risk tolerance and perception is likely to be context specific; Weber et al. (2002) divided contexts into: ethical, financial, health/safety, recreational and social. Computer security and privacy decisions share features of several of these. Like some health/safety decisions, the consequences may be delayed (e.g., information theft) or unnoticed (e.g., malware), the probability of risk may be perceived as low, the actions may be routine (e.g., reading email), and the immediate primary effects (e.g., playing a new game or looking at a photo of a relative) may be recreational. Unlike health/safety decisions, the cost of the threat is not as high (e.g., slow down of machine) or not as imminent (e.g., bot installed on machine). From the social context, the user may experience peer pressure to take some action and be unwilling to admit when something goes wrong. Like ethical decisions, risky actions may be violating laws (e.g., copyright in peer to peer file sharing) or putting others at risk (e.g., bots). Additionally, the perception of cost for avoiding risk may be high (e.g., time spent learning about security), and the right actions may not be obvious (e.g., which security software to install or how best to make settings in browser). Moreover, given the disclosure of industry and government breaches, users may simply feel helpless.

In addition, the work by Davison and Sillence (2010) shows that human users' security related behavior often does not match their stated intentions. Human users often state the intention of being secure online but then perform actions that put their system at risk. Two studies (Govani & Pashley, 2005; *2010 NCSA / Norton by Symantec Online Safety Study*, 2010) have been able to verify that the self reported values for some human user computer behaviors are in fact higher than the actual values. This finding confirms that human users are unable to recognize on their own that they need help while using computers, and would benefit from automated solutions that can help them during the decision making. To this end, developing user-friendly security solutions requires observing users performing actions rather than relying on self reporting, which acts as a proxy for actual behavior. Ideally, the home computer user should be studied at home for an extended period of time to be able to accurately gauge the computer security related behavior patterns. However, these requirements pose many logistical and practical problems. For example, the software system used in such studies should be a background process with minimal intrusion to the user's normal computing activities while collecting the necessary system/behavior information at different levels of granularity (e.g., system calls, process information, user action sequences). Knowledge of having these computer security monitoring software installed in their personal computers may force users to practice computer security more thoroughly than they would normally, which will lead to artificial behavior data. Therefore, we need to balance collecting high quality information against privacy and practicality of studies. The sandbox environment we present in this chapter is designed to address the problem of collecting reliable data from human users when practicing computer security. Using the sandbox environment the subject is asked to walk through a protocol (task list) in a controlled environment, in which they are presented with different decisions to be made and actions taken. The scenarios can be designed to test the effect of cues and responses of users, thus allowing us to study computer security practices without having to only rely on self reports.

3.2 Related Work

Before effective interventions can be developed, we must understand what types of users make particular decisions, and how they prefer to make decisions. What are the factors that influence user's decision making? Studies that have tried to answer this question have used behavioral models as a guide. Like HBM, there are many other behavioral models that have been adopted to study human user behavior such as Theory of Planned Behavior (Ajzen, 1991), Extended Parallel Process Model (Witte, 1992) and Precaution-Adoption Process Model (Weinstein & Sandman, 2002). These models define constructs that collectively represent a person's actual control over the behavior. Studies that apply these models to explain user behavior in computer security have looked at computer security practices such as password management, email usage, backing up data, antivirus software and firewall usage in both home and organizational environments. Typically, the studies evaluate the effects of the model's constructs on practicing security using self reported surveys from human subjects. We now discuss human behavior models that are commonly adopted in studies about the home computer user's security behavior.

3.2.1 Modeling User Behavior for Computer Security

Users adopting computer security concepts share many similarities with how they respond in a health related scenario. For example, consider a user installing an antivirus software to protect his computer against malicious. This is very similar to someone adopting a healthy diet to avoid diseases. HBM was developed in the 1950s after the failure of a tuberculosis screening program attempted to explain and predict health behaviors. HBM consists of six core constructs that affect an individual's core beliefs on some health related concept. These constructs are: susceptibility, severity, benefits, barriers, cues-to-action, and self-efficacy of performing a given health behavior. Let us translate the HBM for a scenario where the user is trying to protect his computer against a virus attack using antivirus software. HBM states that the user's behavior will depend on the user believing that there is a high probability that his computer will be affected by a virus (susceptibility), the user believes that the negative effect of the virus attack is a serious problem for him (severity), the user believes that having an antivirus software is helpful in preventing virus attacks (benefits), the user believes that there is little difficulty in getting a good antivirus software and using it (barriers) ,the user believes that he can recognize external trigger events indicating the computer may be at risk (cues-to-action), the user believes that he is confident in his ability to install and use an antivirus software to protect his computer (self-efficacy).

HBM has been adopted in several studies about computer security practices of human users. Ng et al. (2007) study the effects of HBM constructs operationalized to the security practice of exercising care when reading emails with attachments. They evaluate the effects of HBM constructs using self-reported behavior of users. Unlike our study, which focus on home computer users, this study is conducted in the organizational context. The study finds that when exercising care while opening email attachments, self efficacy, susceptibility and benefits are determinants of user behavior. They further find that barriers, cues-to-action and perceived severity are not significant determinants of security behavior. However, they note that although the cues they have evaluated (e.g., organizational awareness programs) are not effective in prompting secure behavior in users, other types of cues could prove to be effective. In our study, we intend to address this by evaluating cues that can be presented from within the computing environment (e.g., browser notifications, secure URLs) to promote secure behavior in users.

Claar et al. (2010) studies the effects of HBM constructs in the context of using computer security software anti-virus, firewall, and anti-spyware. Like our study, this study also focus on computer security behavior of the home computer user. Claar also extended the HBM by adding moderating variables gender, age, education, prior experience on the effect of computer software usage. Participants were recruited from university students and Internet groups and were administered a survey. The results show that self-efficacy, barriers, susceptibility influence computer security usage. This study also found that cues-to-action did not support how human users practice computer security.

Attack graphs (Ammann, Wijesekera, & Kaushik, 2002) have been proposed as a methodology of modeling how a security vulnerability (goal state) can be reached starting from an initial state through a set of state transitions. Urbanska et al. (2013) propose an extension of the attack graphs that factors in the user's interactions with the system (in addition to the attacker) that leads to the manifestation of vulnerabilities. This state/action model is referred to as a Personalized Attack Graph (PAG). The PAG provides a basis for answering two questions related to trigger identification: (a) Which are the "appropriate" system states that must be monitored? and, (b) Which user actions are predicted to possibly lead to a vulnerability? A PAG explicitly ties the dependencies between known vulnerabilities existing in a standalone system and the system configuration (including characteristics of the user), to the user activities, system actions and attacker actions that can lead to an exploit of a vulnerability. The personalized aspect of the PAG comes from incorporating information about the user as *user attributes*. These attributes include, user actions, preferences and assets. The authors use a Bayesian network model influenced by HBM to assign probabilities to the user attributes. User attribute probabilities for a specific user is calculated from values assigned to the six primary factors in the HBM and six additional factors (prior knowledge, experience, gender, age, socio-economic and education). However, the PAG model is evaluated using synthetic data and expert opinion. In this work, we produce actual human subject data that can be used to accurately estimate these probability values for two of the HBM constructs: self-efficacy and cues-to-action.

3.2.2 Challenges in Security User Studies

Many studies on assessing human decision making in practicing computer security (including the studies cited in the previous section), report that one major challenge in conducting security user studies is the disparities between self-reported and actual observations. Howe et al. (2012) observed that most studies that look into computer security practices of users relying on self reported surveys suffered from issues such as respondent bias, socially desirable responding and peer perception. The authors posited that experiments based on simulation, which place the participant in the actual situation that is monitored can help reduce such issues and also be leveraged to assess the emotional reactions of users to interventions and warnings.

Questions asked in self-reporting surveys induces respondent bias. For example in Ng et al. (2007), the user is asked the question "*I am confident of recognizing suspicious email headers. (agree/disagree)*" to measure self-efficacy. Users may find it difficult to estimate there skill level for some task and often overestimate their abilities. Furthermore, asking about computer security practices primes the user for security, which may be difficult for researchers to control. A human subject study that looked into effectiveness of SSL warnings that appear during Web browsing tasks (Sotirakopoulos, Hawkey, & Beznosov, 2011) attempts to minimize the priming effect by designing tasks for the Web browsers the participants normally used and simulating the same look and feel of the native Web browsers. This study also used deception. The researchers did not reveal the true purpose of the study until the tasks were completed. The results of this study also confirms that there is a disparity between self-reported and observed actions in computer security studies. We follow the same principles in our sandbox environment design and the administering the experiment.

The sandbox environment has the same look and feel as the typical Microsoft Windows Desktop. We adopted deception by presenting the participants with a cover story, which stated that we were evaluating the participants on their ability to perform day-to-day computing tasks like checking email and web browsing. We also use a combination of surveys and observational data to identify predictors of user actions. Surveys are issued to participants at the beginning and at the end of the study to more accurately gauge what the users report and what they had actually done.

While the sandbox offers a controlled environment, it also affords realism and the ability to conduct longitudinal studies. This is particularly important in practicing computer security, which must continue beyond the initial adoption. Our sandbox study only looks into computer security practices in laboratory conditions and the study is administered only once. Accessing a user's computer in an unobtrusive way for a long period of time presents considerable challenges. Warkentin et al. (2016) attempt to address this issue and presents a model that explains continuous intention to practice computer security. The researchers, measured the usage of an anti-malware software over nine weeks. The software was installed in the subjects' computers and usage data was sent to a web based server periodically. The study finds that in order to form an intention to continue the use of security software the effects of threat severity, susceptibility and self-efficacy are significant contributing factors.

Another challenge in human subject studies in computer security practices is the sample bias. Characteristics of the study participants vary from the general population of computer users. Many studies use university students as the subject pool (Sotirakopoulos et al., 2011; Warkentin et al., 2016) in order to draw a representative sample to the best of their ability. However, typical computer users are young, old, tech-savvy, non-tech-savvy, educated, uneducated, male, female and so on. The diversity of the sample should lead to more generalizable data. In our study, we also use the university students as a sample mainly because it is a sample of convenience. We however restricted the student sample to consist of non computer science majors so that we can better approximate an average home computer user.

3.3 PsychoRithm: A Sandbox for Conducting Security Studies of Users

We now present PsychoRithm, the sandbox environment we designed and implemented to study how the self-efficacy and cues-to-action impact security decision making of home computer users. When running an experiment using PsychoRithm, the subject is presented with a desktop, which includes common applications such as emailing, Web browsing and file browsing. As the subject performs these activities, different events happen that can trigger threats and vulnerabilities of interest to the researcher. These events may include asking the user to register with a login name and a password, to provide sensitive information to gain access to a site, to respond to pop-ups asking the user to download a software. PsychoRithm records the user responses to these actions, including user clicks and time taken for response.

We had several design objectives. First, the environment needs to be as realistic as possible to encourage subjects to behave as they would on their home computers. When people are accustomed to using particular software, they have expectations about how to use it and what it will do. These expectations direct how they interact. Second, the software must be portable to different settings while also protecting the computational platform, allowing for simulation of security vulnerabilities without actually causing damage. We wanted a system that could be easily moved to a user location to run experiments. For example, if we wanted to study mature adults, we could take the setup to a senior center and conduct the experiments from there by installing the software on available platforms. Whether we run the study on someone's computer or our own, we need to retain control of the effects of the subject's actions. Yet, the possibility of infecting study machines with security experiments gone awry is very real. Given such constraints, we decided to create a fully simulated environment that would mimic the behavior of real world interaction without making any changes to the underlying operating system and minimizing changes to the file system. For example, if a pop-up asked the user to click on a button to download a software, the user would be able to click on the button, a download progress bar would show the changes, and finally a file icon would be placed in a mocked up file system browser to give the appearance that an actual file has been downloaded. Third, data collection needs to be computationally lightweight and privacy protecting. We considered a variety of methods for monitoring user interaction and found, unsurprisingly, that the most comprehensive are also the most computation intensive and potentially privacy invasive. We needed to strike a balance between carefully collecting/storing valuable data

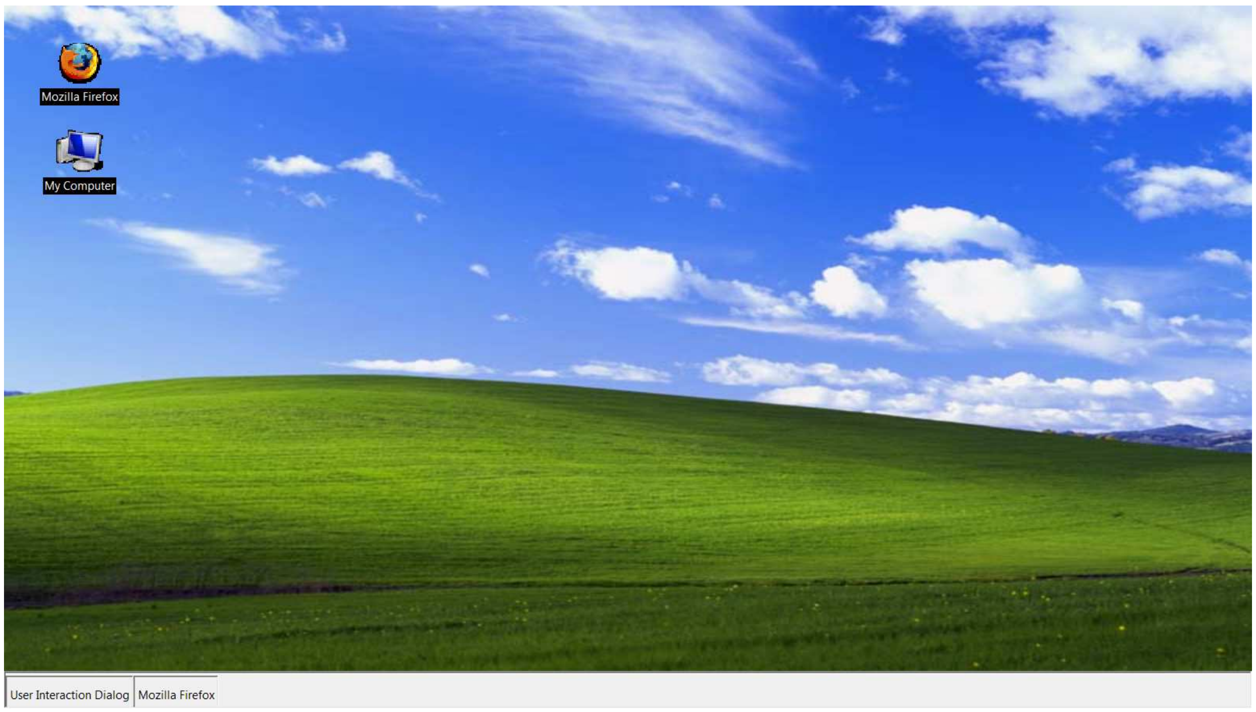


Figure 3.1: Simulated Microsoft Windows desktop environment

and damaging the privacy or the subject experience (e.g., not slowing down the computer so much that the subject notices). We now describe the tool we developed to support our controlled user study by first describing the user view and then the system view.

3.3.1 Simulating Common Computing Activities in a Familiar Context

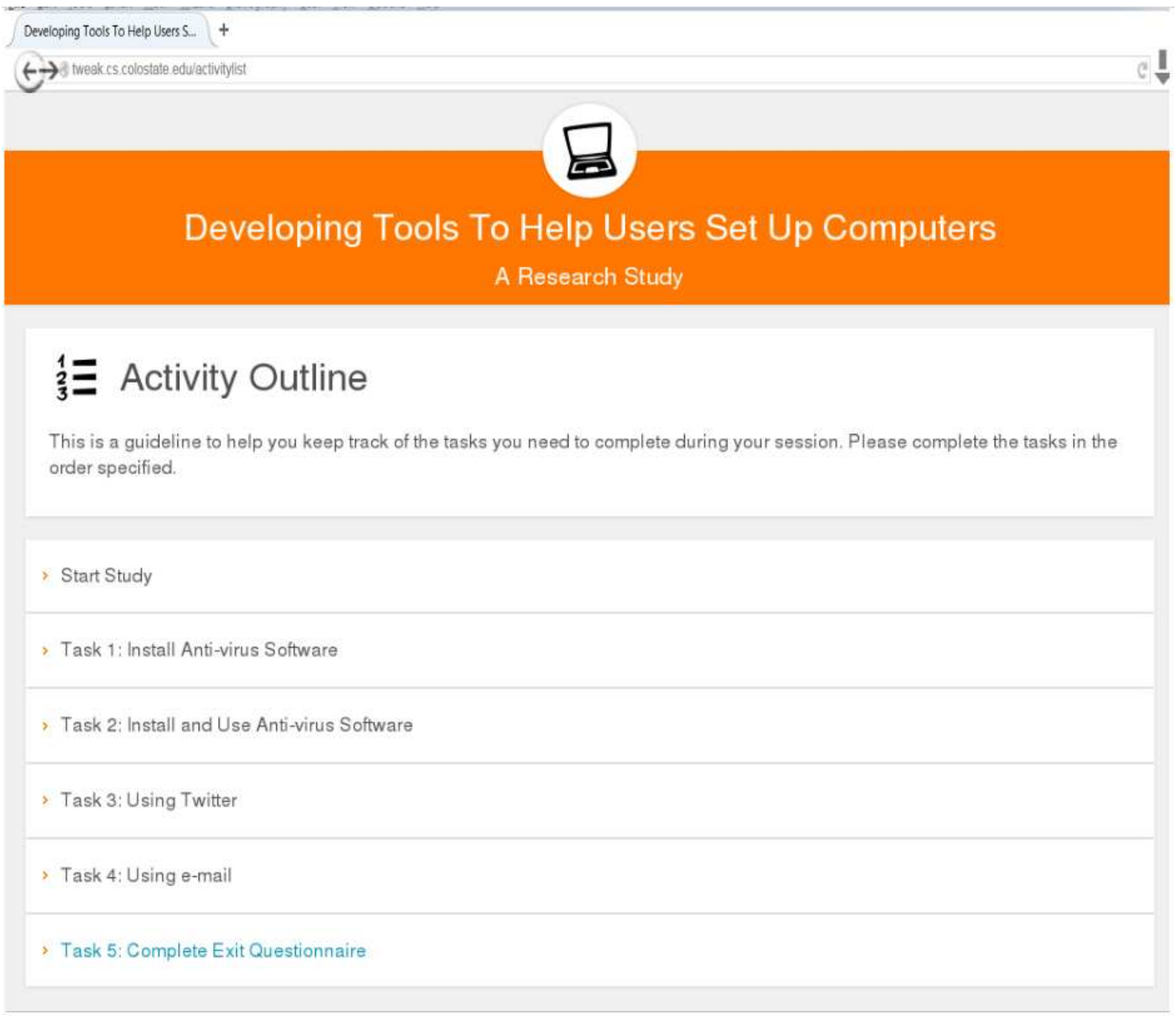
Microsoft Windows is one of the most widely used desktop computing platforms accounting for 88% of the market (Vaughan-Nichols, 2020). Thus, the look-and-feel of the Windows desktop and common applications is important to instilling familiarity for subjects. When a subject starts, he/she sees a Desktop as shown in Figure 3.1. The Desktop provides application icons, e.g., a web-browser and a file-browser. Subjects can click these icons to launch the corresponding applications.

In our study, the browser is the nexus; most of the interaction transpires through the browser. Subjects were instructed to open the browser which showed the starting page for the study (see Figure 3.2). The browser not only guided the subjects through the tasks of the study, but also ensured that the subjects followed the required sequences. Subjects were allowed to perform a task only when the previous task was complete. For example, installing an application could not be skipped as subjects were required to post a verification code, which they obtained after running the application.

For the browser, we used FireFox (version 34.0.5) because it is open source, which allows us to customize the software to enable only the features we need for the simulated environment (forward/backward page navigation buttons, download button, address bar). To maintain the experience as being focused on computer interaction, web pages were used to collect pre and post interaction survey data. For our study, the data included demographic information as well as reports that could be compared against behavior as the subjects executed their tasks. To make the pages realistic, the URL bar in the PHP stub displayed the expected URLs (e.g., https://twitter.com) with an associated padlock to indicate that the HTTP connection was secure, when appropriate. The URL bar could be changed to test subject sensitivity to cues such as these. The browser also allowed for multiple tabs to be open for the experiment landing page and the various applications that are run within the browser.

Studies on global Internet users show that checking email, sending/receiving direct messages (chat), browsing for information/research and social media activities account for the bulk of the online activities (Furnell, Bryant, & Phippen, 2007; of Southern California Annenberg School Center for the Digital Future, 2018). Thus, simulating these activities was a priority for our tool; the current version includes simulated applications of email and Twitter. Because “google and click” is so open ended, we did not include it in the current version of the tool.

Our email application was based off of SquirrellMail (version 1.4.22). SquirrelMail includes built-in PHP support for the IMAP and SMTP protocols and all pages render in pure HTML. These configurations fit perfectly to the technologies behind our simulated environment and therefore we could easily integrate it as an add-on service to the desktop environment. Our version of mail supports login and message functions. Subjects login using provided usernames and passwords and then can change their password. They can look at a list of messages, read and answer them. For purposes of this study, the email accounts can be populated with a set of email messages. Figure


Developing Tools To Help Users S...

tweak.cs.colostate.edu/activitylist

Developing Tools To Help Users Set Up Computers

A Research Study

Activity Outline

This is a guideline to help you keep track of the tasks you need to complete during your session. Please complete the tasks in the order specified.

› Start Study

› Task 1: Install Anti-virus Software

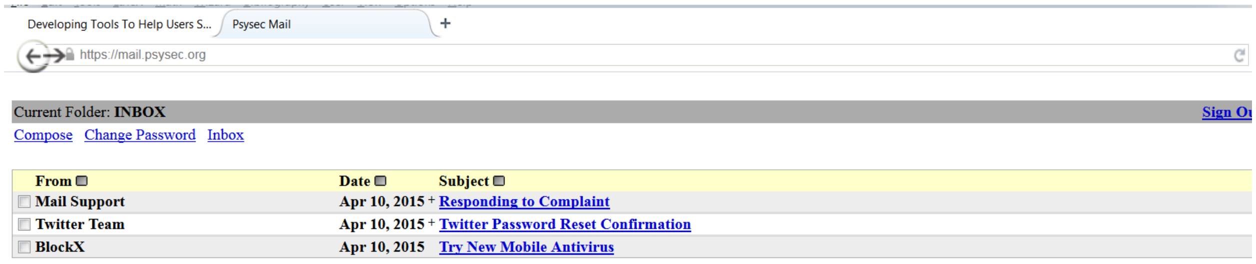
› Task 2: Install and Use Anti-virus Software

› Task 3: Using Twitter

› Task 4: Using e-mail

› Task 5: Complete Exit Questionnaire

Figure 3.2: Simulated Mozilla Firefox Web browser


Developing Tools To Help Users S... Psysec Mail

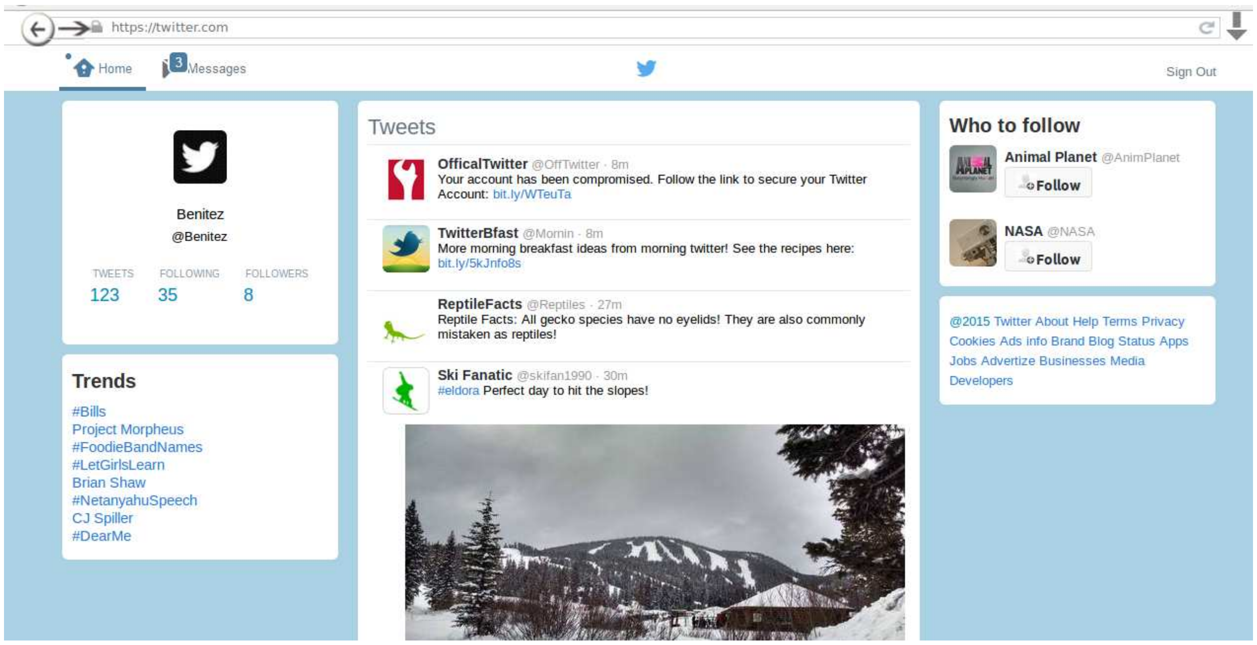
https://mail.psysec.org

Current Folder: **INBOX** Sign Ou

Compose Change Password Inbox

From	Date	Subject
Mail Support	Apr 10, 2015 +	Responding to Complaint
Twitter Team	Apr 10, 2015 +	Twitter Password Reset Confirmation
BlockX	Apr 10, 2015	Try New Mobile Antivirus

Figure 3.3: Simulated SquirrelMail email client


https://twitter.com

Home 3 Messages Sign Out

Benitez @Benitez

TWEETS 123 FOLLOWING 35 FOLLOWERS 8

Trends: #Bills, Project Morpheus, #FoodieBandNames, #LetGirlsLearn, Brian Shaw, #NetanyahuSpeech, CJ Spiller, #DearMe

Tweets

OfficalTwitter @OffTwitter · 8m
Your account has been compromised. Follow the link to secure your Twitter Account: bit.ly/WTeuTa

TwitterBfast @Mornin · 8m
More morning breakfast ideas from morning twitter! See the recipes here: bit.ly/5kJnfo8s

ReptileFacts @Reptiles · 27m
Reptile Facts: All gecko species have no eyelids! They are also commonly mistaken as reptiles!

Ski Fanatic @skifan1990 · 30m
#eldora Perfect day to hit the slopes!

Who to follow

Animal Planet @AnimPlanet Follow

NASA @NASA Follow

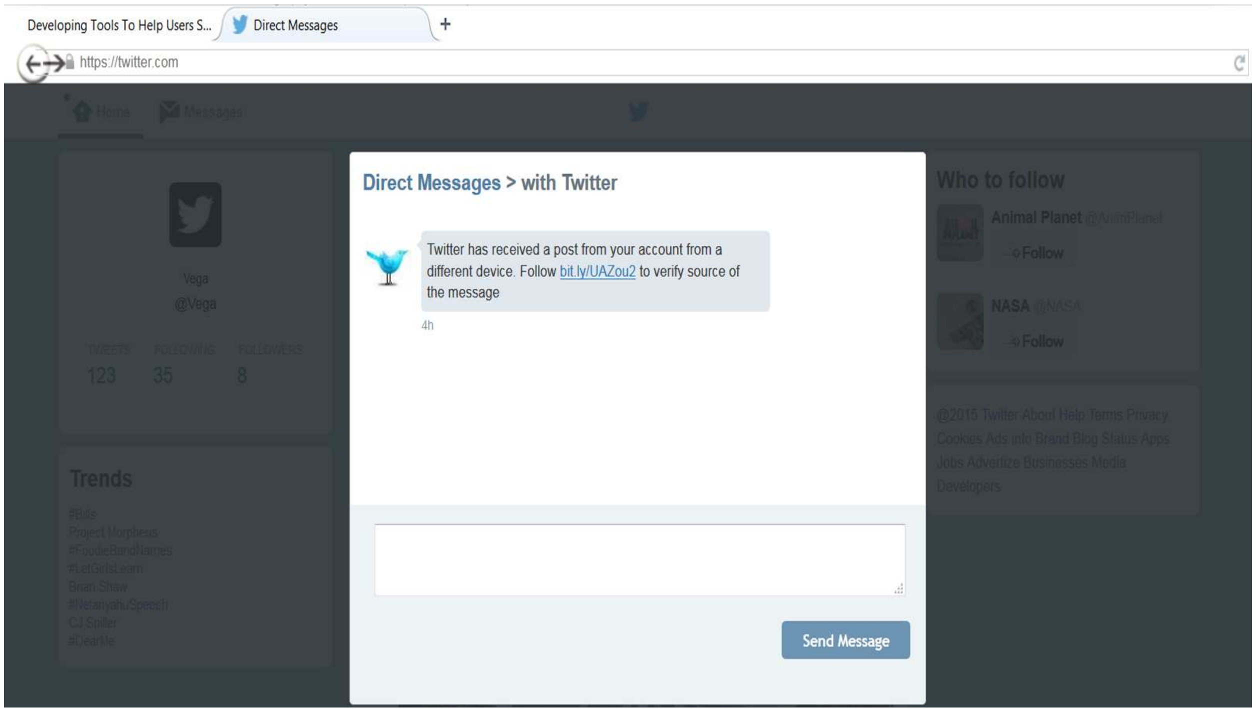
@2015 Twitter About Help Terms Privacy Cookies Ads info Brand Blog Status Apps Jobs Advertize Businesses Media Developers

Figure 3.4: Simulated Twitter home page with three direct messages in the inbox

3.3 shows three emails that were used in our study and that were related to the other applications in the experiment protocol.

As with email, our Twitter application supports login and message functions. Each participating user was provided with a twitter handle and a password. Three messages were posted in their respective Twitter inboxes (see Figure 3.4). One of the three messages as shown in Figure 3.5, which appears to link to a phishing website; if they click on the link, they are taken to a site which requested subjects to provide their Twitter username and password.

To monitor the user's aptitude to use a common software application, we included an anti-virus software selection and installation activity. The subjects accessed an anti-virus download


Direct Messages > with Twitter

Twitter has received a post from your account from a different device. Follow bit.ly/UAZou2 to verify source of the message

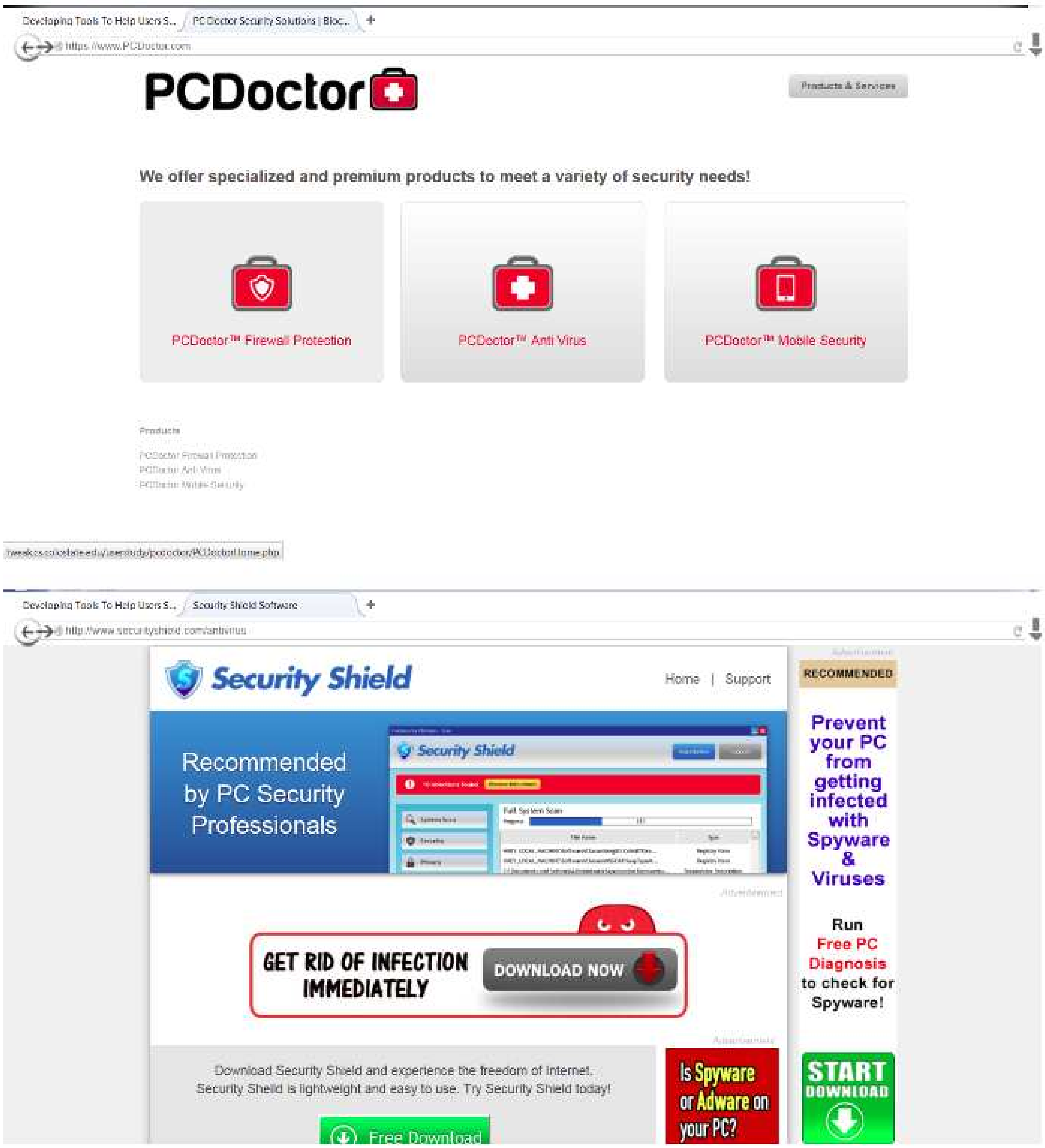
4h

Figure 3.5: Simulated Twitter direct message

portal which leads to two separate websites as shown in Figure 3.6. One of the web-sites was purposefully made secure (HTTPS) while the other was made to look like a typical phishing site with flash images and dramatic warnings. Once the applications were downloaded, the subject was notified of the completed download. The subjects were then required to install and/or run the applications. To make sure the subjects installed and ran the downloaded application, the portal requested a verification code which was obtained by running the downloaded applications. Both anti-virus 'downloads' were equipped with simulated system scanning capabilities (e.g., a progress bar appearing after the system scan is started).

3.3.2 System Architecture

The PsychoRithm software was designed to operate as a kiosk. The software creates a Simulated Local Environment on the subject's machine. The required applications such as the email program, the Twitter site, anti-virus software web pages, phishing web pages, and the survey ques-


Developing Tools To Help Users S...
PC Doctor Security Solutions | Bloc...
https://www.PCDoctor.com
PCDoctor
Products & Services
PCDoctor™ Firewall Protection
PCDoctor™ Anti Virus
PCDoctor™ Mobile Security
Products
PCDoctor Firewall Protection
PCDoctor Anti Virus
PCDoctor Mobile Security
Developing Tools To Help Users S...
Security Shield Software
http://www.securityshield.com/antivirus
Security Shield
Home | Support
Recommended by PC Security Professionals
RECOMMENDED
Prevent your PC from getting infected with Spyware & Viruses
Run Free PC Diagnosis to check for Spyware!
GET RID OF INFECTION IMMEDIATELY
DOWNLOAD NOW
Is Spyware or Adware on your PC?
START DOWNLOAD
Free Download

We offer specialized and premium products to meet a variety of security needs!

Download Security Shield and experience the freedom of Internet.
Security Shield is lightweight and easy to use. Try Security Shield today!

Figure 3.6: Simulated Anti-virus download pages: secure anti-virus software (top) and unsafe anti-virus software (bottom)

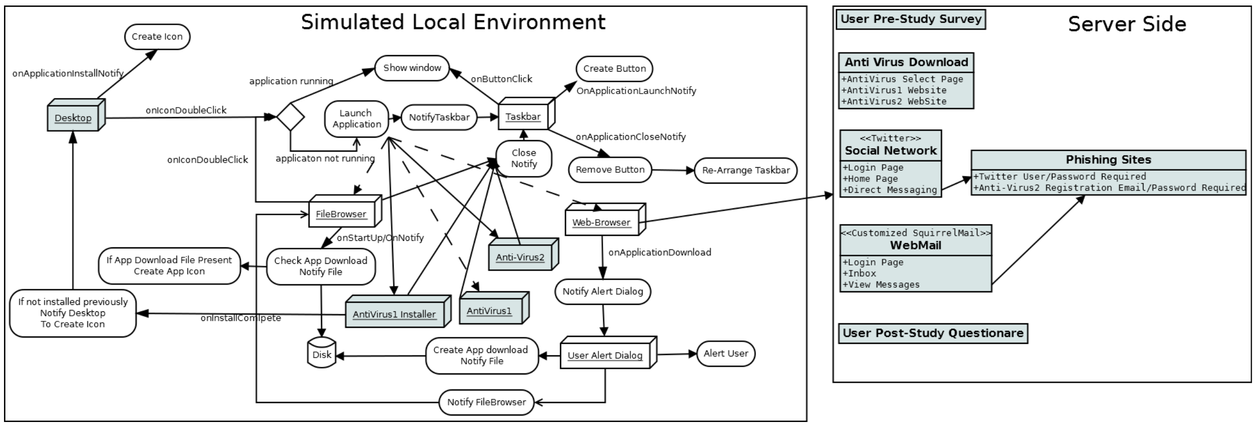


Figure 3.7: PsychoRithm software architecture diagram

tionnaires reside on the server. Software architecture diagram for PsychoRithm is shown in Figure 3.7

Desktop and File System

The user interacts with the desktop and the file system to download, install and run applications. At the start of the experiment, the user validates his experiment username with the desktop, and the desktop creates a folder specifically for that user. The simulated desktop provides application icons: Mozilla FireFox Web browser and the My Computer file browser (Figure 3.1). The user can click these icons to run the corresponding application. When new applications are installed, such as the anti-virus software, new icons are added to the desktop. The desktop and all applications are supported by a taskbar, which provides easy access to all opened applications.

To make the simulated desktop resemble a real desktop, we implemented it as the topmost Win32 Tool Window spanning the working area of the actual desktop. Underneath the simulated desktop is an invisible list-view window, which hosts the application icons. Inter-process communication was performed using custom defined messages with application specific WPARAM values. We implemented our own opaque taskbar as a Win32 Toolbar that maps a zero-based incremental index to the launched applications. This index was used to track buttons/window-handles and close/rearrange them when applications were closed.

PsychoRithm was built as a package that contained empty folders like the *Downloads* and *LoginData*. It was placed under the `C:\Users\<login_name>\Documents}` directory (to avoid conflict with system directories) and identified with `login_name` on the fly to create the absolute path to the package. However, since the path was created dynamically, we could not configure the exact location of the *Downloads* folder into Firefox and hence used the *User Alert Dialog*. A Win32 window with an underlying list-view was used as the file browser for the *Downloads* folder.

Web Browser

The Web browser component allows the user to browse the content fetched from the server. To create the illusion that applications were downloaded from a specific URL, we disabled the Firefox warning dialog by editing the mimeTypes.rdf which led to Firefox calling the User Alert Dialog. Similarly, to make websites like Twitter appear to be served from twitter.com, we removed the URL bars from the existing Firefox application and introduce a custom PHP stub (enabled with Javascript navigation functions) which emulated a URL bar and allowed subjects to navigate/reload pages in same browser session. This required mocking up images that made the interface appear as expected and means that any visited URL must have a corresponding image to be loaded. Action login was facilitated by PHP session variables and AJAX event handlers.

All widgets were disabled using a FireFox extension. When subjects double-clicked on the Firefox icon, the Firefox window was brought above the simulated desktop window giving the illusion that a new application has been launched. A custom designed Firefox extension allowed subjects to minimize this window, but not quit it.

Email

The SquirrellMail portal ran on top of a DoveCot IMAP/POP3 server.

Anti-virus Applications

Two anti-virus applications were created using Java Swing Platform. One of the anti-virus applications requires installation. After installation, it notifies the Desktop, which then creates an icon to launch the application. To make the download process realistic, the web browser makes a call to the *User Alert Dialog*, which then alerts the user, creates a 0 byte file in the Downloads folder and notifies the file browser.

Collecting Data

The system is instrumented to collect subject actions, specifically the start and conclusion of the session, the activation of each activity, typing and mouse clicks in the applications (buttons, widgets, windows, etc.) The results of the pre and post surveys and subject preferences (e.g., software selections, password choices) were also logged using PHP session variables. All activity related data are captured with timestamps. Once the experiment is complete, the Desktop process sends over all captured data to the server side using SFTP/SCP, cleans all of the local storage and kills any process that was initiated by the system.

As per the regulations of the Institutional Review Board (IRB) of the Colorado State University, the researchers are required to store subject data in a safe and secure manner. Moreover, to run enough subjects, we needed to allow multiple subjects to participate simultaneously. Consequently, a secure remote server also stored experiment data in a safe manner. To preserve anonymity, the data were associated with the IDs (login names), which were assigned randomly when subjects arrived for the study and were never associated with specific individuals.

Configuration and Extensions for Other Studies

Microsoft Windows and Visual Studio Framework are required to install and use PsychoRithm. The web applications (Twitter, Antivirus Software Download) pages are implemented using HTML/AJAX/JQuery. The Anti-virus mock-up software is implemented using Java. For this study, the simulator only supports Web browsing (restricted only to predefined URLS), file browsing (restricted only to personalized *Downloads* directory) and email (restricted only to SquirrelMail webmail application) applications. Access to other Windows applications are restricted through the simulated desktop and the web application shown in Figure 3.2.

Because Firefox is an existing application, it must be started at the beginning of an experiment session and cannot be stopped during the course of the experiment. As a result, the subjects were only allowed to minimize the application and restore it via the taskbar button. All menu operations, including right-click menu operations, were disabled. In this study, our main focus is on evaluating users' behavior with regard to self-efficacy and cues-to-action while performing common tasks on the computer. There are other behavior models describing many other factors that affect user's computer security behavior. When evaluating the factors beside the two we are testing in this study, in their real usage context, the simulator software may require extensions for additional, more complicated tasks. In this case, future add-on Windows applications can be developed using the aforementioned technologies and integrated as plug-ins.

3.3.3 Studying Behavior In Situ

The goals of this experiment are:

- capturing actions taken by users when asked to perform tasks that provided "opportunities" to trigger security vulnerabilities
- characterizing their behaviors by their gender and prior experience
- assessing consistency in subjects answers and actions.

In this study our focus was on two factors common to several well-known models of user security behavior: self-efficacy and cues-to-action (e.g., Health Belief Model, Protection Motivation Theory). The core idea was to ask questions related to those two factors, then observe the user actions within scenarios designed to elicit potentially detrimental responses and then to ask questions about the tasks that they had undertaken. This protocol was designed to collect data on how the two factors manifest in the participant sample and to allow us to compare the self reports of against the observed actions. While we had no hypothesis about how the factors would appear, our hypothesis about the comparison of data collection methods is that self-reporting was likely to differ from observed actions. Thus, the study was designed with five parts: an introduction, a pre-study questionnaire, on-line completion of four tasks, a debriefing and a post-study questionnaire.

The sandbox software allowed us to analyze the actual behavior of human users when they are presented with scenarios that may lead to serious computer security exploits such as installing illegitimate software, responding to phishing email, not changing default passwords, potentially in contrast to what they report in a survey. If the participants were directly informed of the real purpose of the study (i.e., computer security practices), it is likely they would be extra cautious in completing the tasks. This self-consciousness could influence what they do in the experiment making it differ markedly from their actions outside the context of the research study (e.g., at home, workplace) and introduce an additional bias to measurements we collect. To this end, we introduced some deception into the study in three ways:

1. The description of the study used to attract participants indicated that the study was to evaluate their ability to complete common tasks on the computer so that the researchers can use the findings to develop tools to help users set up their computers.

2. A task, which was unrelated to the focus of the study (i.e., compose and save email as draft) was added to one of the other tasks to help mask the actual purpose of the study. This task did not generate any useful data.

3. To induce a sense of buy in, the participants were presented with a cover story at the start of the study. The cover story stated that the participant had purchased a second hand computer with Microsoft Windows and one utility application (Mozilla Firefox) installed. Experiment tasks were to study the participants ability to set up this newly purchased computer to be used as an additional computer by multiple users at their home. The cover story was communicated to participants during the introduction session.

During the debriefing session, participants were informed of deception used in this study. The study was designed as a within-group experiment, where all participants were placed in identical threat scenario simulations. Written instructions were given to all participants to assist them in completing the required set of tasks. All data were collected through the PsychoRithm sandbox. Text computer logs recorded the tasks that were completed, timestamp at which a task was completed, the selections made when options were presented, external URLs visited by following links on web pages, and updates made to user visible settings. Video/audio recordings were not used. The study took approximately 45 minutes to complete.

Participants

In Spring 2015, we collected data from 61 university undergraduates as participants. The students were enrolled in a psychology course during that semester. The course attracts a wide range of majors and gives research credit for participating in studies. All participants were given informed consent and participated voluntarily.

Although selected as a sample of convenience, the sample was representative of home computer users from a critical group: young adults with easy access to computers. Prior studies have shown that high school and college aged adults, age between 18 and 34 report the highest level of Internet usage (*Computer and Internet Use in the United States*, 2012). 56% of participants in this study were female. 74% were less than 20 years of age, 25% were aged 20-25 years and the remaining participant was aged 26-30. 98% were working toward a Bachelor's degree.

Surveys

We presented two surveys. The pre-study survey shown in Appendix A asks three questions (questions 1 - 3) about demographics and 10 questions about computer experiences (questions 4 - 13). Although we would have liked to directly assessed self-efficacy, the standard questions, as pioneered by (Ng & Xu, 2007; Claar, 2010), asked about level of confidence when executing specific tasks in different contexts. We did not wish to prime the participants to be thinking about computer security during the on-line portion of the study. Instead, we followed the tone of Mannan and van Oorschot (2008), which asked about their experience with computers (questions 4 - 8), what challenges they perceive (question 9), where they obtain help (questions 10 - 11) and what software they have installed (questions 12 - 13). These questions were selected to relate more clearly to the cover story.

For the post-study survey shown in Appendix B, we adopt the standard survey instruments, which evaluate self-efficacy for computer security practices from (Ng & Xu, 2007; Compeau & Higgins, 1995). Instruments that measure cues-to-action are adopted from (Ng & Xu, 2007). It must be noted that our idea of cues refer to security notifications available in the computing environment such as SSL lock icon in URLs, web site content/look and feel as opposed to external information sources like news, help-desks etc. in Ng et al.(2007). Questions 2 - 11 evaluate the self-efficacy of the anti-virus software usage. Questions 12 - 19 evaluate self-efficacy of safe communication over social media (Twitter). Questions 21 - 24 evaluate self-efficacy of using email and managing passwords. Questions 20, 25-29 are about awareness of secure and private information collection. Questions 30 - 35 evaluate cues-to-action on practicing computer security on the Internet. The first question asks for the pseudonym which allows us to connect the on-line portion's data with the post-survey data. The main purpose of these questions was to establish whether or not the actions in the online portion of the study corresponded to what they thought that they did.

Experiment Procedure

When the participants arrived for the study, one person checked their name on an attendance list, and then another person had them pick a persona ID from folded slips of paper in a basket. The persona ID included a fictitious name, user ID and password to login to the simulated desktop. All data were collected using the persona ID, which was never connected back to the participant's actual name. They were then led to a computer and assisted in logging into the kiosk software.

The starting page provided the instructions to the participants. To avoid alerting them to security issues, we presented the cover story illustrated in Figure 3.8 for what they were being asked to do.

The participants were presented with a screen that described and organized their on-line tasks. This page was generated as the participant finished one activity and moved on to the next. They were first directed to take the pre-study survey. Their last task was to complete the post study

You have just bought a second-hand desktop computer from Surplus Property, the recycle, reuse sales point of Colorado State University. This computer will be used as an additional computer at home. The computer already has Microsoft Windows operating system installed. Notepad, Windows Media Player and Mozilla Firefox web browser are installed as application software. Additionally, you also managed to get an Internet connection setup for the computer. This computer will be used by you and your family members for regular purposes such as checking email, browsing Internet, and also for entertainment activities like watching movies and videos.

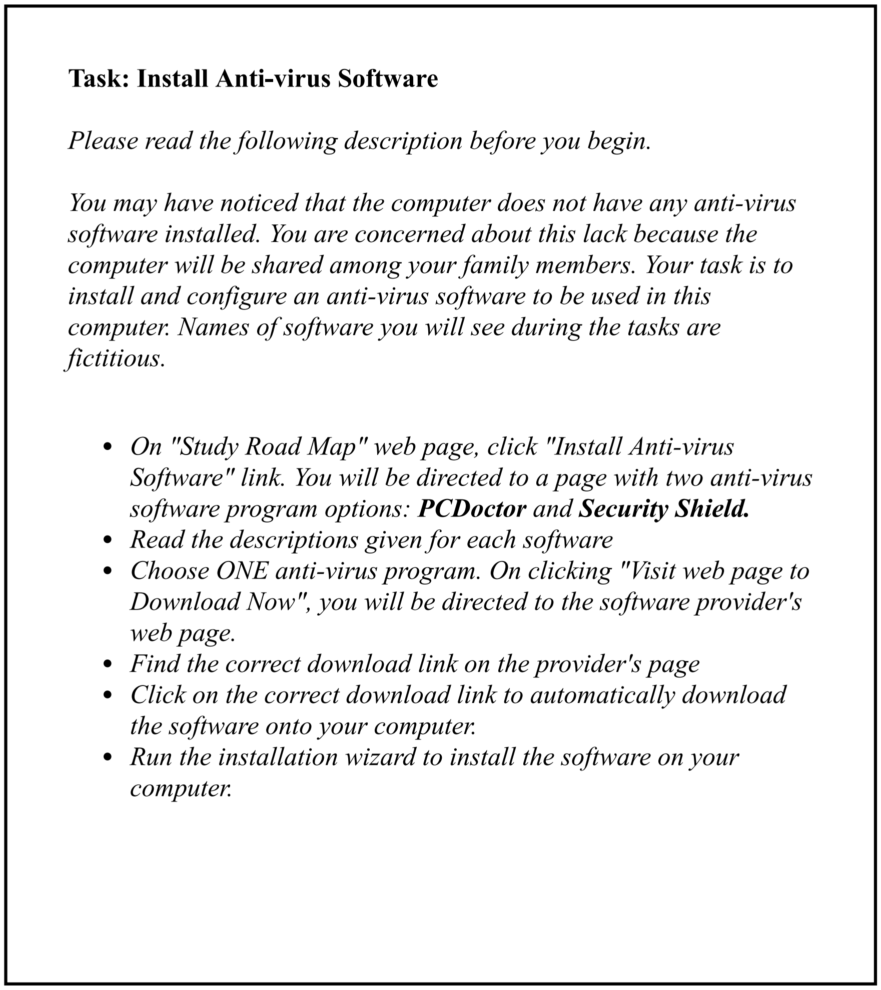
This study helps us understand your ability to perform common tasks on a computer, while it is being set up. To familiarize yourself with the new environment, you may browse the computer and check what applications are installed. Please note that, for the purpose of this study, a custom desktop environment was created and therefore it may not have all options available in a typical desktop computer.

Figure 3.8: Cover story adopted for the cyber security human subject experiment

survey. The online portion presented three threat scenarios: (1) selecting, installing and using an antivirus software to ensure safety of the computer, (2) communicating on social media and (3) using email.

In the first scenario, participants were directed to install, configure and run anti-virus software as part of setting up their new machines. Below is the the instructions that were presented to the user as they started these tasks.

Two antivirus software installations were offered via a web-based download portal implemented within PsychoRithm. One software was designed to be legitimate and safe; the other was intended as suspect and insecure. Neither were existing products or websites; in each case, we patterned our fictitious web pages after exemplars of each type of antivirus software website. The legitimate software is “PCDoctor”, and the suspect one is “Security Shield”. The portal provided a brief description about each software program to assist users make a decision about which program to download. Once the software was downloaded, users were given written instructions to



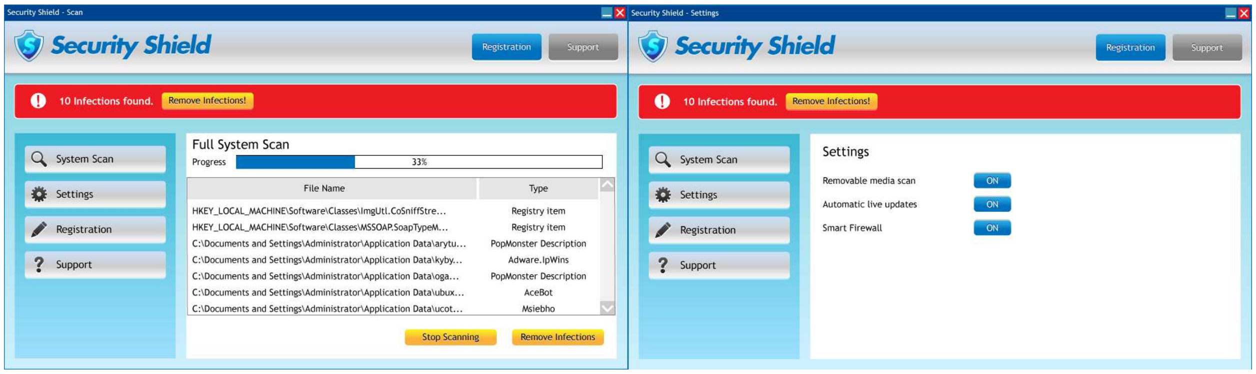
Task: Install Anti-virus Software

Please read the following description before you begin.

You may have noticed that the computer does not have any anti-virus software installed. You are concerned about this lack because the computer will be shared among your family members. Your task is to install and configure an anti-virus software to be used in this computer. Names of software you will see during the tasks are fictitious.

- *On "Study Road Map" web page, click "Install Anti-virus Software" link. You will be directed to a page with two anti-virus software program options:* ***PCDoctor*** *and* ***Security Shield.***
- *Read the descriptions given for each software*
- *Choose ONE anti-virus program. On clicking "Visit web page to Download Now", you will be directed to the software provider's web page.*
- *Find the correct download link on the provider's page*
- *Click on the correct download link to automatically download the software onto your computer.*
- *Run the installation wizard to install the software on your computer.*

Figure 3.9: Description for the install, configure and use anti-virus software task



10 Infections found.

Full System Scan

Progress 33%

File Name	Type
HKEY_LOCAL_MACHINE\Software\Classes\ImgUtl.CoSniffStre...	Registry item
HKEY_LOCAL_MACHINE\Software\Classes\MSSOAP.SoapTypeM...	Registry item
C:\Documents and Settings\Administrator\Application Data\arytu...	PopMonster Description
C:\Documents and Settings\Administrator\Application Data\kyby...	Adware.IpWins
C:\Documents and Settings\Administrator\Application Data\oga...	PopMonster Description
C:\Documents and Settings\Administrator\Application Data\ubux...	AceBot
C:\Documents and Settings\Administrator\Application Data\ucot...	Msiebho

Settings

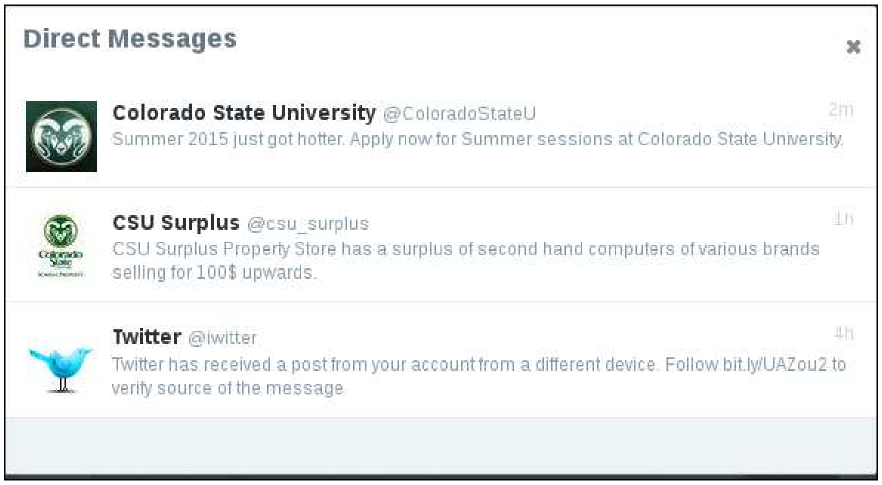
Removable media scan	ON
Automatic live updates	ON
Smart Firewall	ON

(a) System scanning **(b)** Anti-virus protection settings

Figure 3.10: Security Shield anti-virus mock-up software

install the software, configure it to enable three additional security features (automatic live updates, removable media scan, and smart firewall) and finally perform a simple scanning task on the computer. Although neither program actually did anything, both were designed to give the users the interaction experience of an authentic antivirus program with appropriate timing pauses and messages (see Figure 3.10).

A mock-up version of a popular social media application (Twitter) was implemented for the purpose of studying users' communication habits. In this activity, users were instructed to use a private messaging feature of Twitter. In this case, we did create an version of the interface for the specific media site because it corresponded well with the cover story and we were interested in their behavior within this likely familiar context. Users were shown three private messages: two legitimate, safe messages sent from trustworthy sources (different offices within the university at which the study took place) and one phishing message. As shown in Figure 3.11, all messages contained URLs. The URL in the phishing message, directed users to a phishing website `bit.ly/UAZou2`. Once the users visited the phishing website, they were prompted to enter their username and password. In their instructions, they were told that this task would allow them to check that Twitter was working correctly on the computer; they were asked to login, read messages and "take appropriate action".


Direct Messages

Colorado State University @ColoradoStateU 2m
Summer 2015 just got hotter. Apply now for Summer sessions at Colorado State University.

CSU Surplus @csu_surplus 1h
CSU Surplus Property Store has a surplus of second hand computers of various brands selling for 100$ upwards.

Twitter @iwitter 4h
Twitter has received a post from your account from a different device. Follow bit.ly/UAZou2 to verify source of the message

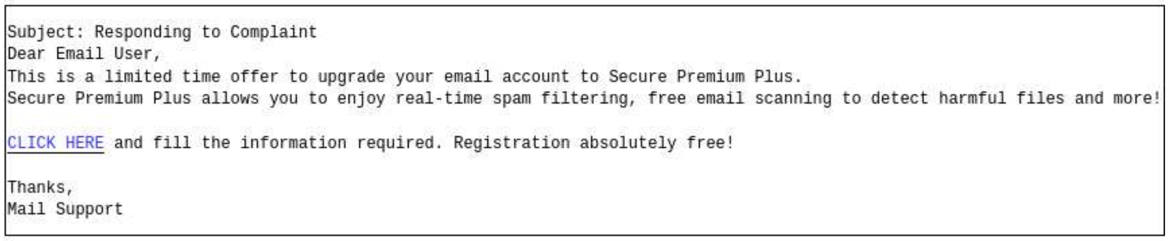
Figure 3.11: A screenshot of the messages as they appeared to the participants

A web based email client was used for the email scenario. Users were given personal email accounts and a common default password designated for the study. They were told "The email account is unique to you. However, the default password is being shared by all users of the computer." The email task comprised two sub tasks. First, to study their ability to choose strong passwords, participants were instructed to change the default password of their email accounts to a new password of their choice. Second, users were instructed to compose a new email message, read email messages and respond to unread messages.

As shown in Figure 3.12, three new messages were placed in the inbox: one safe message and two unsafe messages. One of the two unsafe messages was a phishing message with a URL that would direct the user to a phishing website upon clicking. The phishing website was masked as the web-mail client itself. The other unsafe message contained an attachment: a harmful zip file. The message content was patterned after actual phishing email messages. The phishing websites and zip file attachments were mock-ups within the sandbox; thus, they did not pose an actual security threat to the computer or to the participant's personal data.

We designed the phishing website to follow examples we had seen before and also placed several cues in line with commonly recognized properties of online phishing attacks (*How to Recognize and Avoid Phishing Scams*, 2019) for both the social media phishing attempt and email phishing attempt. First, in the case of Twitter, although the message sender's name was "Twitter", the

Message 1 (unsafe - contains a link to a phishing website)

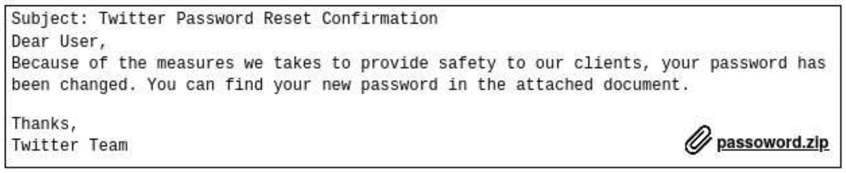


Subject: Responding to Complaint
Dear Email User,
This is a limited time offer to upgrade your email account to Secure Premium Plus.
Secure Premium Plus allows you to enjoy real-time spam filtering, free email scanning to detect harmful files and more!

CLICK HERE and fill the information required. Registration absolutely free!

Thanks,
Mail Support

Message 2 (unsafe - contains malicious zip file attachment)

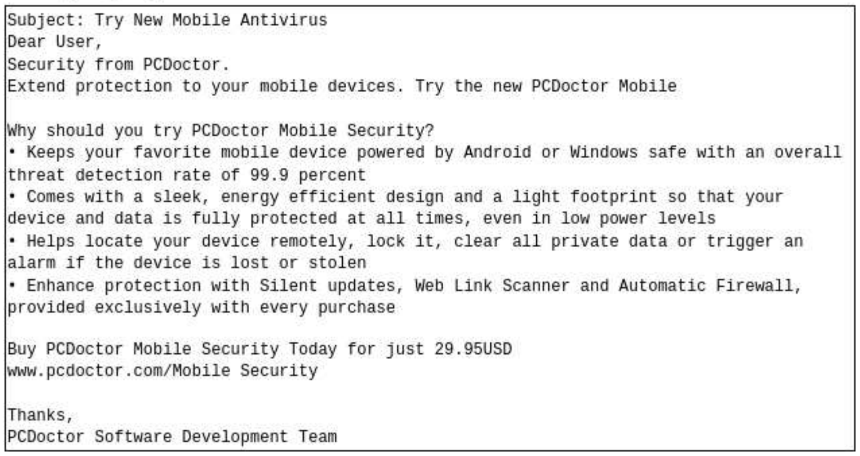


Subject: Twitter Password Reset Confirmation
Dear User,
Because of the measures we takes to provide safety to our clients, your password has been changed. You can find your new password in the attached document.

Thanks,
Twitter Team

passoword.zip

Message 3 (safe)



Subject: Try New Mobile Antivirus
Dear User,
Security from PCDoctor.
Extend protection to your mobile devices. Try the new PCDoctor Mobile

Why should you try PCDoctor Mobile Security?
• Keeps your favorite mobile device powered by Android or Windows safe with an overall threat detection rate of 99.9 percent
• Comes with a sleek, energy efficient design and a light footprint so that your device and data is fully protected at all times, even in low power levels
• Helps locate your device remotely, lock it, clear all private data or trigger an alarm if the device is lost or stolen
• Enhance protection with Silent updates, Web Link Scanner and Automatic Firewall, provided exclusively with every purchase

Buy PCDoctor Mobile Security Today for just 29.95USD
www.pcdoctor.com/Mobile Security

Thanks,
PCDoctor Software Development Team

Figure 3.12: Email messages presented as part of the online portion of the study.

Twitter handle (i.e., message sender's twitter account name) was not similar to the legitimate Twitter account's Twitter handle. The legitimate account's Twitter handle is `@Twitter`. The phishing message was sent from the Twitter handle `@iwitter` (i.e., Twitter misspelled). Second, the message said that some log-in attempts have been noticed. Third, the message content requested that the recipient confirm some personal information, specifically the user's Twitter account user name and password. If the user failed to recognize these cues and clicked on the link, the user is directed to a web page, which contains a log in page but looks different from the legitimate Twitter login page. Users had already seen the legitimate login page at the start of the Twitter task and ideally should be able to recognize that the current web page is different. We added several cues to this page as well. We changed the Twitter logo and the color scheme of the phishing web page. The URL did not contain the SSL lock icon and was `http://iwitter.com`, whereas the legitimate web site's URL was `https://twitter.com`. If the user did not recognize the phishing web site given all these cues and submitted the user name and the password, the user was directed to a landing page, which displayed a message that the user's personal information has now been compromised. Then the user was directed to the next task.

A similar approach was taken to handle security attacks that occur via email. The phishing message on email was masqueraded to appear as if it was being sent by the Anti-virus software vendor. Recall that, in the sequence of tasks, when the user reaches the email task, they would have already completed the anti-virus task and is familiar with the Anti-virus vendor's name. We embedded the same discrepancies between the legitimate vendor's information and the phishing website as the Twitter phishing scenario as cues for the email phishing use case. If the user clicked the phishing link, the same sequence of events took place as in the case of the Twitter phishing scenario. There were several cues to recognize the malware email such as the generic greeting from the sender ("Dear User"), an enticement to open an attachment (get the new password by opening the attachment), information review check (password change) and undisclosed recipient ("Mail Support"). If the user clicked the attachment, the user was directed to a web page, which displayed a message that the computer system's safety is now compromised.

Once the participant finished the tasks, he/she was directed to the post-study survey and a debriefing explanation. We also offered them the opportunity to further discuss the debriefing with one of the staff conducting the study.

3.4 Data Analysis and Results

This study has two main components that evaluate users' computer security behavior: self-report surveys and the sandbox activities. Therefore, we divided the analysis as within component (only surveys and only sandbox) and across component (surveys and sandbox). We first present the analysis on single components.

3.4.1 Pre-Survey Findings

The survey can be found in the Appendix A, page 294. Table 3.1 shows a summary of the pre-survey findings. Although the participants had a lot of prior experience with computers (66% with more than 9 years), they had little formal training (59% had none) and turned to the Internet for help as their primary source (92%). Most (82%) owned multiple devices. Nearly all (95%) had installed multiple applications, i.e., Chrome, Office, Flash, Skype and Acrobat. Given the participant recruitment method, it was not surprising that nearly all had laptops and used their computers for educational activities. With respect to computer security, more than half (67%) found it challenging to identify harmful actions and take steps to ensure safety. Yet most (77%) had anti-virus software on their computers.

Gender and Computer Usage Patterns

We analyzed the data to identify any differences that may appear between male and female participants. For questions that could be coded as numbers, we ran two sample, two tailed T-tests. For questions that had categorical answers, we constructed $2 \times R$ contingency tables, where R was the number of categories, and ran Pearson's chi-squared tests. All analyses were performed using the R statistical package.

Table 3.1: Summary data from the pre-study survey questions. Each row is a question with its number and short description. Highest values in each row are in bold.

Years experience	4-6: 11%, 7-9: 23%, **>9 : 66%**
Usage hrs/day	<3: 30%,**3-6: 54%**, 7-10: 11%, >10: 5%
Devices	**laptops: 97%**, smartphones 82%, tablets: 33%, desktop PCs: 21%, e-Readers: 7%, Other: 3%
Applications	**web browser: 98%**, Office: 82%, Acrobat: 61%, Media Player: 49%, Computer Games: 26%, Image Processing: 21%, Software Development: 11%, Other: 3%
Purposes	**Education: 93%**, Documents: 84%, Communication: 82%, Entertainment: 82%, Information gathering: 74%, Financial: 56%, Work: 41%, Games: 26%, Programming: 10%
Challenges	**Identifying actions that may harm the computer: 67%**, Taking steps to ensure the safety of the computer: 51%, Installing, configuring and getting a software application ready to be used: 43%, Finding application software that matches my requirements: 39%, Finding and using help manuals: 25%
Training	**None: 59%**, Application Courses: 21%, Programming Courses: 13%, On-line Tutorials: 5%, Face to Face Classes: 2%
Help sources	**Internet: 92%**, Friends: 54%, Relatives: 46%, Local Stores: 23%
Anti-virus usage	**Yes: 77%**, No: 23%
Installations	**Google Chrome: 89%**, Office: 85%, Flash: 75%, Skype: 62%, Acrobat: 57%, Firefox: 33%, QuickTime: 25%, VLC Media Player: 21%, Thirdparty email: 10%, Other: 7%, Opensource Office: 3%

Table 3.2: Comparing male and female participants on questions coded as numbers, Rows show mean for females, mean for males and p value for T-test.

	Yrs Experience	Usage hrs/day	Devices	Applications	Challenges	Installations
Mean Female	8.88	4.81	2.50	3.55	2.68	4.59
Mean Male	9.07	4.20	2.33	3.70	1.70	4.74
p-value <	0.66	0.33	0.53	0.69	0.002	0.73

Table 3.2 shows the results for gender on questions 4, 5, 6, 7, 9 and 13 (page 294). We coded years of experience (question 4) as 5 for 4 - 6 years, 8 for 7-9, and 10 for $>$ 9. We coded hours/day (question 5) as 2 for $<$ 3, 4.5 for 3 - 6, 8.5 for 7 - 10 and 11 for $>$ 10. For questions 6, 7, 9 and 13, we counted the number of answers given by each participant. Although the means on each of these questions shows some differences between female and male participants, that difference is significant for only one question: the number of challenges. The female participants reported 2.68 on average, where the male participants reported 1.70. Thus, the perception of self-efficacy in computer usage for the females was clearly lower than for males.

Table 3.3: Results from Pearson's Chi-Squared test on gender for questions 6 - 13. Columns show χ-squared value, degrees of freedom (df), p value and the category with the largest % difference between females and males (negative indicates lower value for females. $\alpha < 0.05$

Question	χ	df	p <	Largest Difference
Devices	5.58	4	0.23	Desktop PC -21.6
Applications	5.69	6	0.46	Computer games -26.0
Purposes	8.44	8	0.39	Playing games -26.0
Challenges	6.68	4	0.15	Identifying harmful actions 47.5
Training	12.55	4	0.01	No training 39.4
Help sources	6.29	3	0.10	Relatives 29.2
Anti-virus	4.10	1	0.04	Yes 25.3
Installations	9.50	9	0.39	VLC -21.6

Questions 6 - 13 had categorical answers. For these, we used each of the pre-specified answers as categories. We had few cases in which "Other" was used and so did not include that as a separate category. Table 3.3 summarizes our findings. The last column lists the category that showed the largest percentage difference and lists the percentage after the category name. For installs, no females and only 2 males installed OpenOffice, so we did not list it as the largest difference.

Only two questions showed possibly significant differences due to gender in the chi-squared tests. Participants reported a considerable difference in formal training: 76.5% females had no formal training compared to 37% of the males. Anti-virus installation was more common for females (88.2%) versus 62.9% for males. One hypothesis could be that less formal training leads people to take more action with respect to security. To see whether the association might be due to the lack of formal training, we constructed a 2×2 contingency table of Training (Yes/No) versus Antivirus Installation (Yes/No). A chi-squared test yielded P < 0.637 which suggests that the effect is not likely due to the lack of formal training. As a next step, we examined whether lack of formal training leads to more perceived challenges. To test this, we ran a two sample T-test on number of challenges for those with no training versus those with some training. The mean for no training was 2.47 and with some training was 1.92; the p-value from the T-test was 0.08, suggesting a marginal effect. Gender appears to exert more of an effect than training on the perception of challenges.

Table 3.4: Percentage of responding and visiting link for Twitter tasks. Also rate of submitting password for the phishing message.

Message	% Responded	% Visited Link	% Submitted Password
Safe Message 1	41.0	59.0	N/A
Safe Message 2	32.8	57.4	N/A
Phishing Message	19.7	75.4	68.3

3.4.2 Sandbox Findings

Each of the scenarios included multiple tasks. We did not use the software to force participants to complete tasks. Therefore some of the participants skipped certain tasks. Only 30 participants completed the antivirus software installation task. For this task, 76.7% of the participants chose the questionable anti-virus software, only 1 participant clicked on the EULA and 46.7% of participants successfully removed all infected files by running the anti-virus software correctly.

For the Twitter task, the participants were asked to read and possibly respond to three messages. 61 participants preformed parts of the task. As shown in Table 3.4, participants were most likely to visit the phishing link and least likely to respond to the phishing message.

We examined whether there was a significant difference in responding to messages, visiting links and submitting passwords between the safe and phishing messages. To do so, we constructed 2×2 chi-square tables of yes/no to action and pairs of messages as the other factor. We found significant differences in response and visiting rates ($P < .0001$). For the response rates, if a subject did not respond to a safe message, they were highly unlikely to respond to a phishing message (for the first safe message, 0 of 26 did so; for the second, only 1 of 40 did so); those that responded were equally likely to respond to either safe or phishing messages (13 out of 25 and 9 out of 20, respectively). For the visiting links, the relationship is still significant but the pattern is different. If the subject visited the link in the safe message, they were very likely to visit the link in the phishing message (33 out of 35 and 34 out of 24); if they did not visit the link in the safe message, they were equally likely to visit in the phishing message (14 out of 26 and 15 out of 26). The pattern of responding suggests that participants who are cautious about responding do not respond

Table 3.5: Percentage of participants responding, visiting link and downloading for Email tasks.

Message	% Responded	% Visited Link	% Downloaded Attachment
Phishing Message	19.7	73.8	65.6
Twitter Reset	N/A	N/A	29.3
Safe Message	N/A	46.3	N/A

to anything; others pick randomly. The pattern of visiting suggests that participants always click links or chose randomly whether to click.

The safe messages did not ask for their passwords. So we compared following the links in the safe messages to submitting passwords. Again, the relationship was significantly different ($P < .0001$). In both cases, if they followed a link in a safe message they were likely to submit their password; otherwise they were unlikely to submit their password.

For the email task, participants were asked to change their password, compose a new email message and respond to messages. Two of the three messages were unsafe. 41 participants executed this task. As Table 3.5 shows most participants did take unsafe actions in these scenarios.

3.4.3 Post-Survey Findings

Tables 3.6, 3.7, 3.8, and 3.9 show summaries of the post-survey answers for the Antivirus task, Twitter task, Email task and general security questions, respectively. Questions 2 - 24 and 30 are answered on a 5 point scale and are encoded from 1 to 5 (higher is better). For questions 3 - 11, 13 - 24 and 30, from Strongly Disagree to Strongly Agree. For questions 2 and 12, the answers varied from Very Hard to Very Easy. Questions 25 - 29 allowed answers of “Yes”, “No” and “I don’t know”. Questions 33 and 35 allowed “Yes” and “No”. Questions 31, 32, and 34 had categorical answers.

As shown in Table 3.6, for the antivirus task, the highest level of confidence was in configuring and using the antivirus software. The lowest was in removing suspicious files without help, but even here, they still felt more confident than not (values greater than 3 which is neutral). Questions 6 and 7 distinguish ability with and without help. A paired sample T-test comparing the responses to questions 6 and 7 shows a significant difference ($p < 0.014$), which suggests that many partic-

Table 3.6: Summary data from the post-study survey questions concerning the antivirus task. Each row is a question with its number and short description. Questions 3 - 7 assess confidence. Numbers are means followed by standard deviations in parenthesis. Values closer to five indicate high confidence or very easy.

2. Difficulty with installation and configuration	3.623 (1.186)
3. Configuration and usage	3.869 (0.939)
4. Selecting software that matches requirements	3.623 (0.934)
5. Identifying legitimate software	3.525 (0.993)
6. Identifying and removing suspicious files	3.410 (0.990)
7. Identifying and removing suspicious files without help	3.131 (1.126)
8. Checking that download site is trustworthy	3.820 (0.958)
9. Choosing software that is customizable	3.623 (0.879)
10. Using software that is configured to preferences	3.492 (1.043)
11. Importance of software for identifying and removing files	4.098 (0.907)

Table 3.7: Summary data for the post-study survey questions concerning the Twitter task. Each row is a question with its number and short description. Numbers are means followed by standard deviations in parenthesis

12. Difficulty in responding to direct messages	4.361 (0.708)
13. Confidence in using the direct messaging feature	4.311 (0.827)
14. Confidence in identifying suspicious messages	4.115 (0.858)
15. Confidence in identifying suspicious messages without help	3.984 (1.025)
16. Confidence in identifying suspicious messages even if it is the first time	3.967 (0.875)
17. Checking links for an unknown sender or suspicious information	3.918 (1.053)
18. Practicing caution in following links	3.984 (0.904)
19. Not following links if the content looks suspicious	4.098 (0.943)

ipants were concerned about their ability to perform the task on their own. Similarly, questions 4 and 9 address different aspects of selecting antivirus software: whether the software matches requirements and whether it can be customized to do so. A paired sample T-test comparing the answers to the two yield $P < 1.0$ The highest variance was in how difficult it was to install and configure the software. The lowest variance was in choosing software that is customizable. However, the variances were all approximately 1.

As Table 3.7 shows, the participants are a little more confident in their use of Twitter than the antivirus tasks. The highest level of confidence was in responding to messages; the lowest was in checking links. Paired sample T-tests comparing the responses to questions 14 with 15 and 14 with 16 show no significant difference ($P > .09$).

Table 3.8: Summary data for the post-study survey questions concerning the Email task and 5 point scale Security questions. Each row is a question with its number and short description. Numbers are means followed by standard deviations in parenthesis

20. Security warnings stop web site visits	3.639 (0.932)
21. Practicing caution in responding to unknown senders	4.000 (0.913)
22. Practicing caution in downloading attachments from unknown senders	4.033 (0.856)
23. Changing passwords when prompted	4.115 (0.819)
24. Confidence in choosing a strong password	4.213 (0.878)
30. Helpful descriptions	3.721 (0.933)

As Table 3.8 shows, the participants are confident in their use of email. The highest level of confidence was in choosing a strong password; the lowest was in the security warnings. Paired sample T-tests comparing the responses to questions 21 with 22 shows no significant difference ($P > 0.71$).

Table 3.9 summarizes answers from the general security questions. Although the majority indicated they could recognize secure pages and verify secure sites, they were less sure about whether the Twitter page was secure or whether they had downloaded the antivirus software from a secure site. The cue most recognized for legitimate pages was “HTTPS” in the URL which suggests they have some idea of its meaning. They also were aware of the significance of an unrecognizable address in harmful email. The most common reason for not reading EULAs was they are too long.

3.4.4 Between Pre and Post Surveys Findings

By looking across components, we could examine 1) whether their reports of self-efficacy in the pre-survey related to their actions or their post-survey responses and 2) whether participants are consistent in their responses and actions. The pre and post surveys were designed to inquire into the participants’ computer skills and perceptions of computer security and what they do generally and did within the experiment. PsychoRithm was designed to determine what they did when presented with scenarios.

In the pre-survey, question 9 asked about what tasks they found most challenging. We analyzed the data to whether specific self-reported challenging tasks translated into lowered confidence in executing the tasks in the experiment. For example, did participants who reported that finding

Table 3.9: Summary data for the categorical post-study survey questions concerning security behavior generally. Each row is a question with its number and short description. Answers are provided with percentages.

25. Recognizing secure pages	**Yes: 67.2%**, No: 11.5%, I Don't Know: 21.3%
26. Was Twitter page secure?	**Yes: 42.6%**, No: 26.2%, I Don't Know: 31.1%
27. Concerned about safety	**Yes: 77.0%**, No: 14.8%, I Don't Know: 8.2%
28. Verify secure site	**Yes: 65.6%**, No: 23.0%, I Don't Know: 11.5%
29. Antivirus from secure site	**Yes: 45.9%**, No: 14.8%, I Don't Know: 39.3%
31. Recognizing legitimate pages	Appearance: 60.7%, Contact info: 56.5%, **HTTPS: 73.8%**, Related content: 55.7%, No ads: 54.1%, Popular: 63.9%, Do not know: 8.2%
32. Recognizing harmful email	**Unrecognizble address: 86.9%**, Unsafe links: 83.6%, Bad grammar: 70.5%, Unpersonalized: 63.9%, Odd attachments: 72.1%, Threats: 75.4%, I Don't know: 0%
33. Read EULA usually	Yes: 19.7%, **No: 80.3%**
34. Why not read EULA?	**Too long: 47.5%**, Too technical: 6.6%, No harm: 4.9%, No choice: 19.7%, Not relevant: 1.6%, Did read it: 19.7%
35. Read EULA for anti-virus	Yes: 21.3%, **No: 78.7%**

Table 3.10: Possibly significant relationships between self-reported challenges in pre-survey and self-reported confidence in post-survey

Challenge	Post-survey Question	P
Installation	Identifying suspicious messages in Twitter w/o help	0.006
Using help	Identifying legitimate antivirus software	0.008
Using help	Removing files using antivirus software	0.009
Using help	Removing files using antivirus software w/o help	0.002
Identify harmful	Removing files using antivirus software	0.005
Identify harmful	Removing files using antivirus software w/o help	0.007

software was a challenge also have lower confidence in how they performed the antivirus task? For the analysis, we treated each task as a binary, either the participant reported it as a challenge or did not. We then ran two sample T-tests comparing those who did view it as a challenge to those who didn't for post-survey questions 2 - 7 (anti-virus), 12 - 16 (Twitter) and 24 (email). Due to the number of comparisons, we report only those with $P < .01$ as possibly significant differences in Table 3.10.

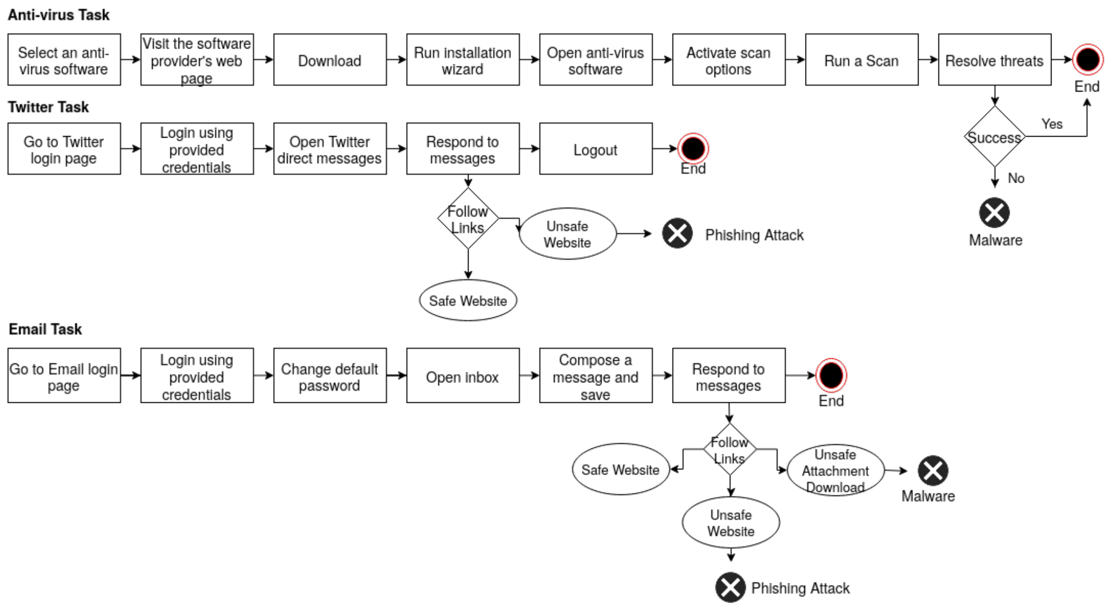


Figure 3.13: Sandbox task breakdown

3.4.5 Between Sandbox and Post-Survey Findings

We now compare the users' actual sandbox activities against what they reported in the post-study survey for the three tasks: using anti-virus, communicating on Twitter and email communication. The post-study survey can be found in the Appendix B, page 297. Figure 3.13 illustrates the task breakdown for the sandbox scenarios. Although the illustration is shown as a sequence (complies with the script that was given to the participants at the beginning of the study), we found that some participants did not follow the scripted sequence. They skipped a sub-task to come back to it later. Other times, the participants tried the same sub-task several times.

Self-efficacy

The anti-virus task consisted of three sub-tasks: (1) given a legitimate and an illegitimate antivirus software, the user was asked to choose one (choice), (2) install the selected software and configure it to be used in the computer (install) and (3) run the software on the computer and remove threats (scan). In the post-survey, question 4 measures self-efficacy of the choice, question 9 measures the self-efficacy of the install and question 6 measures the self-efficacy of the scan. From the 62 participants only 11 completed all three sub-tasks, while 32 participants skipped the anti-virus task altogether. Those who did not complete the task skipped the install or scan or both. All 11 participants who completed all three sub-tasks installed the legitimate anti-virus software. This shows that users who practice computer security typically execute similar actions when they are placed in similar situations.

Listed below are the three questions we provided for the participants to subjectively rate their efficacy to use Antivirus software. Within parenthesis next to each subjective rating value is the numerical score we assigned for that particular rating. When computing the mean efficacy for an activity in the Completed group, we sum the numerical scores of the subjective ratings given by the participants and divide by the number of participants in the Completed group who have answered that question. For example assume three participants in the Completed group rated the efficacy of the Choice task as [*Neutral*, *Disagree*, *Agree*]. Then, the mean efficacy for the Choice task in the Completed group is $\frac{3+2+4}{3} = 3$. The mean efficacy for an activity in the Incomplete group is calculated similarly.

- (**efficacy of choice**) I feel confident in my ability to select an antivirus software that matches my security requirements.

  Strongly Disagree (1) Disagree (2) Neutral (3) Agree (4) Strongly agree (5)

- (**efficacy of install**) I choose an antivirus software, which allows me to customize its features to match my security requirements.

  Strongly Disagree (1) Disagree (2) Neutral (3) Agree (4) Strongly agree (5)

- (**efficacy of scan**) I feel confident in my ability to identify and remove suspicious files on my computer using an antivirus software without help.

  Strongly Disagree (1) Disagree (2) Neutral (3) Agree (4) Strongly agree (5)

Table 3.11 shows the result of the two-tailed T-test that compares the self-reported efficacy between those who completed the choice task and those who did not is significant ($P < 0.05$). The same can be observed for the scan activity. However, the self-reported efficacy is not significant for the install

Table 3.11: Self reported self-efficacy differences between those who completed all anti-virus sub-tasks and those who did not complete the task.

Activity	Completed Efficacy Mean (SD)	Incomplete Efficacy Mean (SD)	P
Choice	4.181 (0.603)	2.944 (0.998)	0.00096
Install	3.6 (1.055)	3.428 (0.851)	0.636
Scan	4.0 (0.707)	2.875 (1.147)	0.005

task. This shows that for certain tasks that involve anti-virus software usage, users can incorrectly assess their self-efficacy when practicing computer security in self-reports.

Our post survey contained 3 questions (questions 14 - 16), which evaluated self efficacy of communicating safely on Twitter. As seen in Figure 3.13 the Twitter task contained the least number of steps. Furthermore, the action "Respond to messages" is a repetitive action. The action makes the user to read and respond to the three messages in their Twitter inbox. All 61 participants who completed the task opened/responded to at least one of the three messages. Listed below are the three questions we provided for the participants to subjectively rate their efficacy to use Twitter safely. Within parenthesis next to each subjective rating value is the numerical score we assigned for that particular rating.

- (**Question 14**) I feel confident in my ability to identify suspicious messages on Twitter.

  Strongly Disagree (1) Disagree (2) Neutral (3) Agree (4) Strongly agree (5)

- (**Question 15**) I feel confident in my ability to identify suspicious messages on Twitter without help.

  Strongly Disagree (1) Disagree (2) Neutral (3) Agree (4) Strongly agree (5)

- (**Question 16**) I feel confident in my ability to identify suspicious messages on Twitter even if it is the first time I had seen one.

  Strongly Disagree (1) Disagree (2) Neutral (3) Agree (4) Strongly agree (5)

Table 3.12 shows the result of the two-tailed T-test that compares the self-reported efficacy between those who visited the phishing web page following the link that appeared on their Twitter direct

Table 3.12: Self reported self-efficacy means (standard deviation in parenthesis) between those who visited the phishing link and did not visit the phishing link during the Twitter task

Question	Mean (SD) Visited phishing link	Mean (SD) Did not visit phishing link	P
14	4.31 (0.87)	4.04 (0.85)	0.298
15	4.13 (1.20)	3.93 (0.96)	0.571
16	4.31 (0.87)	3.84 (0.85)	0.075

Table 3.13: Self reported self-efficacy means (standard deviation in parenthesis) between those who visited the phishing site and did not submit their passwords and those who visited the site and submitted their passwords while using Twitter. P value is the two-tailed T-test

Question	Mean (SD) Visited-No Password	Mean (SD) Visited-Yes Password	P
14	4.25 (0.9)	4.02 (0.85)	0.335
15	4.25 (0.9)	3.90 (0.97)	0.316
16	4.25 (0.9)	3.80 (0.84)	0.305

message and those who did not. We compute separate p-values for each question that asks the user to rate their self-efficacy. It can be seen that the participants evaluated their self efficacy to recognize phishing attempts while communicating on Twitter higher than it actually was. There is no significant difference between the self-efficacy ratings participants who clicked the phishing link and the participants who did not, $P > 0.05$ for all questions in Table 3.12.

A similar observation can be made for the situation where the participant visited the phishing site and submitted their password information when using Twitter. Questions 14, 15 and 16 are also used to collect the self-efficacy subjective ratings for this task. Table 3.13 shows that although there is no statistically significant difference between self reported self-efficacy ratings for the participants who visited the phishing site and also submitted their password and those visited the site but did not submit the password. The phishing task has different risk levels. Visiting the phishing website has a lower risk level compared to visiting the website and submitting the password. Users who submitted the password to the phishing site further compromised their security by enabling the attacker to steal private information.

Users found it difficult to correctly assess their self-efficacy in practicing caution while communicating with email. Questions 21 and 22 in the post-survey asked users to rate their self efficacy. Within parenthesis next to each subjective rating value is the numerical score we assigned for that particular rating.

- (**Question 21**) I practice caution when responding to email from senders I am not familiar with.

  Strongly Disagree (1) Disagree (2) Neutral (3) Agree (4) Strongly agree (5)

- (**Question 22**) I practice caution downloading attachments in emails from senders I am not familiar with.

  Strongly Disagree (1) Disagree (2) Neutral (3) Agree (4) Strongly agree (5)

When encountered with an email containing a phishing link, two tailed T-test showed that there is no statistical significance between self-efficacy ratings given by users who did not follow the phishing link compared to the users who did ($P > 0.05$). All who visited the phishing link, with the exception of one participant, submitted their login username and the password to the phishing web site. The users who submitted their login username and the password had a mean self-efficacy rating 3.89 (SD=0.90). Similar observation can be made from users who downloaded malicious attachments that come with email. Two tailed T-test revealed that there is no statistical significance ($P > 0.05$) between the self-efficacy ratings given by users who downloaded malicious attachments from email against the users who did not.

Cues-to-Action

We focus on cues that are available on the computer system to evaluate how cues-to-action affect the user's computer security behavior. For the anti-virus usage task, we focused on two cues: (1) whether or not the download page was secure (has HTTPS with SSL lock icon), and (2) software provider/web page appears to be legitimate (look and feel, content etc.). Secure download was tested with post-survey question 29 and legitimate provider cue was tested with post-survey question 8. The Twitter task cues were tested with questions 17, 18, 19, 26. We considered cues such as: (1) unknown sender, (2) suspicious message content, (3)shortened URL links that appear in twitter messages and (4) secure HTTPS Twitter login page. For the email task we used two cues: (1) secure HTTPS web pages and (2) Firefox expired certificate warning. The post-survey questions 20 and 28 evaluated whether or not users picked up on these cues for email.

Listed below are the questions we provided for the participants to subjectively rate their ability to detect cues-to-action when using the Twitter application. Within parenthesis next to each subjective rating value is the numerical score we assigned for that particular rating.

- (**Question 17**) Before following a link on Twitter, I first check if it was sent by an unknown sender or contains suspicious information.

  Strongly Disagree (1) Disagree (2) Neutral (3) Agree (4) Strongly agree (5)

- (**Question 18**) I practice caution when I follow links sent to me on messages on Twitter as they may expose me to a security threat.

  Strongly Disagree (1) Disagree (2) Neutral (3) Agree (4) Strongly agree (5)

- (**Question 19**) I do not follow links sent to me on Twitter messages, if the content looks suspicious.

  Strongly Disagree (1) Disagree (2) Neutral (3) Agree (4) Strongly agree (5)

- (**Question 26**) Was Twitter.com page you saw during the experiment a secure web page?

  Yes No I do not know

Listed below are the questions we provided for the participants to subjectively rate their ability to detect cues-to-action when using email.

- (**Question 20**) When I see a message warning me about the security of a web site (e.g., expired certificate, http:// instead of https://) I am about to visit, I do not visit that web site.

  Strongly Disagree Disagree Neutral Agree Strongly agree

- (**Question 28**) When you submit private information (passwords, email addresses) to websites, do you verify that the site is secure and your information is safe?

  Yes No I do not know

In the anti-virus task, we used user responses to question 8 to compare self-reports of ability to recognize safety of anti-virus software by looking at the provider's web page.

- (**Question 8**) Before choosing a source to download software, I first check if it is hosted by a trustworthy provider as it may expose me to a security threat.

  Strongly Disagree (1) Disagree (2) Neutral (3) Agree (4) Strongly agree (5)

Participants' mean self-report value was 3.86 (SD=0.94) for the safe software and 3.00 (SD=1.00) for the unsafe software. Two tailed, T-test between the users who selected the safe software vs. the unsafe software showed that the mean difference is significant ($P < 0.05$). This allows us to conclude that the web page content cue helps users to select a safe anti-virus software. Listed below are is the question we provided for the participants to subjectively rate their ability to detect cues-to-action to recognize secure web sites when downloading software from the Internet. The specific cue we are checking is the HTTPS URL with the padlock icon on the browser address bar.

- (**Question 29**) During the experiment did you download the antivirus software from a secure web page?

  Yes No I do not know

Question 29 had categorical responses. We used the Chi-square test of independence to determine whether the HTTPS cue is associated with users selecting a safe software. Results showed we can reject the null hypothesis that the HTTPS cue is not associated with users selecting safe software (P=0.002, df=2).

In the Twitter task question 17 evaluated the unknown sender cue. Question 18 evaluated the shortened URL links in the message cue. Question 19 evaluated the suspicious message content cue. Question 26 evaluated the HTTPS with SSL padlock icon cue. Two tailed T-test for ability to recognize cues for questions 17, 18 and 19 between participants who visited the phishing link and participants who did not were not significant (P > 0.05). Question 26 contained categorical responses. We used the Chi-square test of independence to determine whether the HTTPS cues is associated with user with users following unsafe links on Twitter. Results showed we can not reject the null hypothesis that the HTTPS cue is not associated with users following unsafe links on Twitter (P=0.84, df=2).

In the email task question 20 evaluated the cue of the expired certificate warning generated by the browser when following a link. Question 28 evaluated the HTTPS cue, when submitting personal information to web sites. Two tailed T-test for ability to recognize cues for questions 20 between participants who followed a link to visit an unsafe website and those who did not was not significant. Participants' mean self-report value was 3.47 (SD=0.77) for those who visited the unsafe site and 3.82 (SD=0.88) for those who did not visit the unsafe site. Question 28 contained categorical responses. We used the Chi-square test of independence to determine whether the HTTPS cues is associated with user with users submitting personal information to web sites. Results showed we can not reject the null hypothesis that the HTTPS cue is not associated with users submitting personal information to websites (P=0.85, df=2).

3.5 Discussion

The results of our study validate the conclusions in previous studies, which states that human users' security related behavior often does not match their stated intentions (Davinson & Sillence, 2010; Govani & Pashley, 2005; *2010 NCSA / Norton by Symantec Online Safety Study*, 2010). For the software installation tasks, the Twitter task and the email task, the participants overestimated their efficacy to recognize undesirable consequences. This shows that the home computer users do not think about the post conditions of the actions they execute in the computer and therefore can benefit from an intervention system that help them identify actions leading to undesirable consequences. Furthermore, the cues-to-action we have discussed in this study are preconditions that exist in the computer system. For example, identifying whether the web page the user is visiting contains the HTTPS SSL padlock icon, or whether the user is opening an email sent by an unknown (not in address book) sender are preconditions that can be checked by the human user before completing the task (downloading software, clicking a link on the email). Although we expect human users to recognize the preconditions (cues) leading to undesirable consequences all the time, this study concludes that it is not the case. We found that the users can recognize certain preconditions that exist in the system (e.g., the web page that they are visiting to download a software is secure) to help them avoid undesirable consequences. However, in other situations they miss these cues (e.g., recognizing unsafe web pages on Twitter) and as a result may end up compromising security. Therefore, the design of software to assist home computer users in avoiding security and privacy vulnerabilities should be motivated by careful studies of their behavior and perceptions.

PsychoRithm sandbox environment allowed us to place human users in simulated scenarios that compromised their security/privacy and record their actual responses and at the same time capture the self-reported data on how users perceive their confidence in identifying risky situations (self-efficacy) and recognize cues that indicate danger (cues-to-action). Although, we are measuring factors that affect the computer security behavior of users (extracted from the Health Belief Model), our objective in this study is not to validate the model, but rather highlight that regardless the perceived self-efficacy and availability of cues, human users can still end up compromising their security. When the threat is immediate, if the user is not interrupted they may end up putting their safety at even more risk. To this end, the intervention models presented in this dissertation target helping human users avoid undesirable consequences.

In this study we mainly looked at three common home computer user tasks:

- Using anti-virus software to secure the computer
- Communicating with Twitter direct messages
- Using email

By analyzing the results, we identified important conditions that should be addressed when designing an intervention system to help human users. First, even though users were given step-by-step instructions for completing these tasks, they rarely follow the script. The users repeated the same action multiple times, while some actions were omitted altogether. The implication of this observation is that intervention systems that monitor users to help them avoid unsafe consequences need to handle noisy and/or missing actions in the observations. Furthermore, any partially complete task (i.e., a sequence of actions) that has been safe upto now may quickly turn unsafe. This means that the intervention decisions need to be made online. Users perform a large number of actions; while few of them incur much risk, some are pivotal in triggering vulnerabilities. For example, from our using email task, opening email action itself is not a risky action. However, when the email is from an unknown sender and contains suspicious attachments, preconditions to a malware vulnerability have manifested within the system. In this state, opening email is actually high risk action. This means that intervention models need to search for multiple ways for accomplishing the same task. Modeling the problem as a state/action transition system using automated planning allows us to accomplish this requirement. When the problem is modeled as a planning domain, automated planners can be used to generate diverse plans, to sample the space of possible plans that can lead to one or more vulnerabilities and help estimate the probability that particular actions lead to triggering the vulnerabilities.

Second, the objective of the intervention system will be to identify when a user's action(s) may be complicit in triggering the threats and help the user avoid the undesirable state (security or privacy breach). Given a fine-grained model of user and attacker actions and states of the system as an automated planning domain, the intervention system should be able to identify actions that can lead to triggering a vulnerability. A previous study (Byrne et al., 2016) showed that users pay more attention to the benefits of the activities than the risk; they have goals that they want/need to achieve and are willing to take the risk to achieve them. Many triggering actions may be normal activities (e.g., reading email, clicking on links) with the user more focused on the goal than on the risk. Thus, the undesirable consequence recognition problem needs to take into account the user's intention as well as the undesirable consequence. Undesirable consequences prediction is assessing whether an action is likely to lead to undesirable consequences, immediately or as part of an action sequence, or may provide conditions that an attacker can exploit. User intentions recognition determines what the user is intending to do through his/her actions, the short or long term goals. The intervention decision involves a trade-off in early recognition of potential danger of an action versus the certainty that the effects will occur. In the chapters that follow we examine how different metrics captured from the planning problem representation can be used to assess that trade-off. To help the user trade-off benefits versus risks, we also need to know what the user thinks he/she is trying to accomplish. In some cases the goal may be explicitly stated in terms of actions and post-conditions. In other cases, the user's goal may be unknown. We consider both these conditions in the intervention models we present.

Third, personalization becomes most important when the intervention system must decide how to prevent an imminent security/privacy threat or decide what to do when one has been detected. Simply preventing a user from doing something they want to do will undoubtedly result in the security software being disabled. Our underlying hypothesis is that security will be more effective when the user is a partner with the system in the security interventions. In Chapter 6 of this dissertation, we present an intervention model designed for human users that uses automated planning to guide the user away from the undesirable consequence by giving hints.

3.6 Concluding Remarks

The design of software to assist home computer users in avoiding security and privacy vulnerabilities should be motivated by careful studies of their behavior and perceptions. In this work, we created sandbox software to simulate, in a protected environment, the user interface to an operating system and browser so that we could capture subject actions related to security and privacy in a realistic setting. Each subject took an initial survey to assess demographics, computer self-efficacy and perceptions of security/privacy, then used the simulator to "setup and test basic functionality in a recently purchased computer" (the cover story) and then took a survey to assess their perceptions of what they had done. In our analysis, we found that the users (adults taking a psychology course in Colorado State University) are unaware when they have made poor decisions and have triggered vulnerabilities. In the following chapters we present intervention models designed to automatically detect when such situations occur, i.e., that a user's action(s) may be contributing to triggering a security or privacy threat. To identify pivotal events, the appropriate system states and user actions must be monitored and compared to a model of actions that can lead to triggering a vulnerability. We use automated planning to model the actions in environments where the user's decision making context differs in terms of what the user knows about the domain and the type of observational data available to identify pivotal events.

If an action that contributes to a trigger is suspected, it should be put in the context of what the user is trying to do and an estimate of how likely the action is to actually trigger the vulnerability. Therefore, future studies for developing tools to help human users practice cyber-security must employ surveys, sandbox simulations and lightweight in situ monitoring to accurately understand the human users' decision making context. While the sandbox offers a controlled environment, in situ affords realism and the ability to conduct longitudinal studies. Therefore, lightweight tools need to be developed for unobtrusively monitoring the user and logging actions related to security. Longitudinal evaluation of intervention techniques is important to determine whether the new techniques will eventually be susceptible to habituation (a major issue with current intervention techniques) and to examine if user's actions change over time as security awareness increases.

In this study, we only analyzed two specific user traits: self-efficacy, and cues to action when we presented the user with choices of anti-virus software to install and asked them to do ordinary tasks like reading/sending email and reading/responding to tweets. Future studies need to extend the state/action model of the home computer environment by including more scenarios and attacker actions. Although self-efficacy and cues to action capture important aspects of the user's decision making context in the home computing environment: post-conditions and pre-conditions of actions the user executes respectively, there are many other factors that can define the decision making context of a human user (e.g., computer experience, decision-making psychometrics, risk aversion, perceived social status, the need to belong). Developing realistic scenarios that capture these additional traits to study human user behavior in security situations is a challenging yet critical step in developing more personalizable intervention models.

Chapter 4

Intervention by Recognizing Actions That Enable Multiple Undesirable Consequences

In this chapter, we discuss the first intervention model for an observer who intervenes to help the user avoid multiple undesirable consequences. We introduce the **Undesirable Consequence Recognition Function**, that uses three domain-independent metrics to measure the importance of an observed action towards contributing to the undesirable state. The dimensions of the Intervention Problem presented in this chapter are summarized in Table 4.1.

In Chapter 3, we showed that many security threats such as software downloads and phishing require user action or at least acquiescence. The threats are triggered because the user did not understand the consequences of reaching the undesirable state or was not aware of the undesirable state from the start. The implication of this finding is that the observer needs to recognize on time that the plan the user is following will have an undesirable consequence. In Chapter 2, we discussed Plan Recognition as the problem of inferring the goal(s) of an actor by constructing a plan from a sequence of actions or observations. To recognize actions that may lead to undesirable consequences, we propose a variant on plan recognition, where the undesirable consequences prediction problem is viewed as a form of reverse planning; it leverages plan graphs from an existing planner to project possible undesirable outcomes and then trace back to actions critical to their oc-

Table 4.1: Dimensions of the Critical Action Recognition Problem

Dimension	**Domain Specific Properties**
Actors in the environment	User, Attacker, Observer
Goals hidden to the observer	User's goal is hidden Multiple known undesirable states
Types of observations	The observer observes the user's actions The observer only observes the effects of the attacker's actions
Noise in observations	Has missing and extraneous observations
Intervention recovery	Block the recognized undesirable action

currence. We formulate the Undesirable Consequence Recognition problem as a planning problem of three domain independent, weighted metrics: **Timeliness**, **Certainty** and **Desirability**. Against an ideal baseline, we examine the trade-offs in choosing the correct intervention point by varying the weights associated with the metrics, the observability and the noise in observations.

4.1 Background

A wealth of literature in plan and goal recognition has examined how to infer a single agent's plan (C. Geib & Goldman, 2009; Ramírez & Geffner, 2009), the agent's goal (Ramírez & Geffner, 2011; Yin, Chai, & Yang, 2004), or the goals or plans of multiple agents (Banerjee, Kraemer, & Lyle, 2010; Kaminka, Pynadath, & Tambe, 2002). Recent work has even examined plan/goal recognition in the face of noisy observations (C. W. Geib & Goldman, 2005; Vattam & Aha, 2015) or extraneous actions (Gal, Reddy, Shieber, Rubin, & Grosz, 2012; Sohrabi, Riabov, Udrea, & Hassanzadeh, 2016). Yet relatively little of this research considers the question of when to intervene if one wants to thwart the plan or goal, for example, if the agent we want to disrupt is at the risk of producing an undesirable state. The decision of when to intervene must be made judicially. Intervening too early may lead to wasted effort chasing down false positives, helpful warnings being ignored as a nuisances, or leaking information for the next attack. On the other hand, intervening too late may result in undesirable consequences.

Our motivating application is a software agent that monitors the state of a personal computer to help home computer users avoid security and privacy vulnerabilities. Home computer users are viewed as the "weakest link" in computer security because they lack the time, knowledge and resources to defend against the increasing incidents of computer security and privacy threats (Sasse, Brostoff, & Weirich, 2001). Moreover, some threats, such as software downloads and phishing, require user action or at least acquiescence (Howe et al., 2012). Security analysis for home computer users focuses on identifying threats on a personal computer by modeling attacker actions, the system state of the computer and computer user's actions, including both ordinary and risky actions. The Personalized Attack Graph (PAG) extends the attack graph model (Sheyner, Haines, Jha, Lippmann, & Wing, 2002) to support individual computer/user level granularity and by representing the states and actions in PDDL (Urbanska et al., 2013). Attacker actions do not include actions that cannot be observed on the home computer. System states can include levels of partial compromise (e.g., access to the password key-chain), configuration information (e.g., operating system), and state changes achieved on specific system components (e.g., implanting a keystroke logger).

In this work, we examine how well an algorithm can determine the best intervention point for the cyber-security application, calling it a "*critical trigger action*" because it may lead to an undesirable state. As each situation may have a unique definition of the ideal intervention point, we formulate intervention as a combination of three domain independent metrics: (1) **Timeliness**, which quantifies how soon the undesirable state may occur, (2) **Certainty**, which highlights frequently occurring actions as important, and (3) **Desirability**, which measures the contribution of the action to user's own objective. The Desirability metric helps separate common harmless actions that further the user's actual goal from harmful actions to be avoided. A critical trigger action is a user action that maximizes the linear combination of these three metrics.

Given a PDDL domain model, a set of undesirable states to avoid, and an observation trace of actions, our algorithm identifies critical trigger actions. An intervention is correct if, compared to a ground truth trace, the algorithm (1) ignores extraneous actions (i.e., as true-negatives) and, (2) identifies actions leading to undesirable state (i.e., as true-positives). We begin with a study of traces taken from a human subject study for computer security. We then generalize our results to four benchmark planning domains and consider the impact of missing and extraneous observations of actions and test the algorithm. Across all benchmark domains, the Certainty and the Desirability metrics perform well in ignoring extraneous actions, while the Timeliness metric and it's combinations with the Certainty and the Desirability metrics perform well in identifying true positives. Thus, in this work we have identified two metrics that are sensitive to noise in action based observation traces and a metric that is sensitive to partial observability of actions.

4.2 Intervention in Cyber-security Domain

Before introducing a formal definition, we present an example to clarify the key concepts. Suppose a user wants to achieve the goal G (e.g., `read email`) but there exists an undesirable state U (e.g., `installation of malicious software`). Figure 4.1 illustrates the *possible* manifestation of U. The user has executed actions a_1, a_2, a_3 (e.g., `start email application, enter username/password, open inbox`) from the initial state S_0 in order to achieve G. Actions a_1, a_2, a_3 have resulted in the post conditions S_1, S_2, S_3 respectively. An observer is monitoring the actions executed by the user and must intervene upon recognizing that the user has executed an action that will lead to U.

In order to intervene, the observer considers possible plans that lead to U. We denote a plan as a sequence of actions followed in brackets by the undesirable state that results $(a_i, a_j, \ldots, [U])$. We will call these *undesirable plans* identified as Π_U and denote a plan in the set as π_U. Ideally, the user would continue toward achieving G and not execute plans in Π_U. However, by mistake or undue influence, the user may deviate from pursuing G and unwittingly follow paths in Π_U. In the example in Figure 4.1, the observer has seen the action sequence $O = \{a_1, a_2, a_3\}$ the user has executed. In state S_3, the user may reach U through a set of undesirable plans $\Pi_U = \{(a_5, a_8, a_7[U]), (a_5, a_8, a_6, a_{14}[U]), (a_4, a_{10}[U]), (a_4, a_9, a_{16}[U]), (a_4, a_9, a_{12}, a_{11}, a_{13}[U]),$ $(a_4, a_9, a_{11}, a_{12}, a_{15}, a_{16}[U]), (a_{12}, a_{13}[U]), (a_{12}, a_{15}, a_{16}[U]), (a_{12}, a_{15}, a_{11}, a_{12}, a_{13}[U]),$ $(a_{12}, a_{15}, a_{11}, a_{12}, a_{15}, a_{16}[U])\}$. The actions in the plans in Π_U are the **candidate trigger actions**. A candidate trigger action is an action that appears in at least one plan in Π_U. The observer's question is: *given the current state S_3, which action in the set of candidate trigger actions is the best action to intervene if observed?*

4.2.1 The Intervention Problem: Recognizing Undesirable Consequences

Given a domain $\mathcal{D}$ and a planning problem P, an automated planner generates the planning task $\Pi = \langle \mathcal{F}, \mathcal{A}, I, G \rangle$, where $\mathcal{F}$ is a finite set of grounded propositions, $\mathcal{A}$ is the finite set of grounded actions instantiated from the operator schemata Op, I is the grounded propositions specifying the

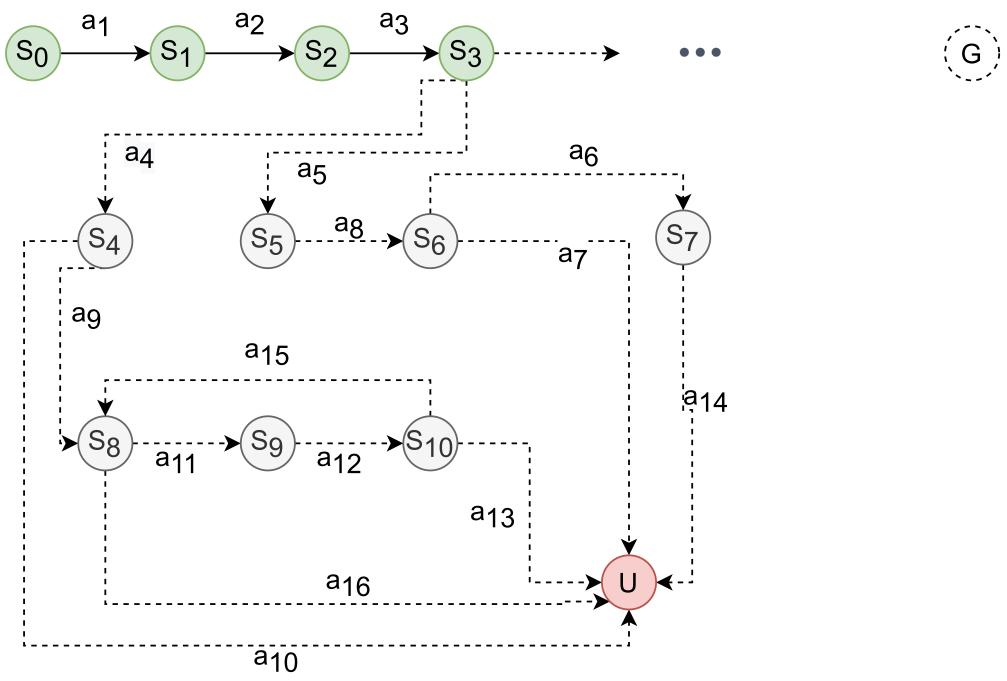


Figure 4.1: An application domain where a user executes the actions a_1, a_2, a_3 to transform initial state S_0 to a goal state G. An undesirable state U may develop from state S_3 after a_3 has been executed, following the plans (Π_U) indicated by the dotted paths towards U.

initial state and G is the grounded goal specification. Objects in $\mathcal{O}$ are used to ground $\mathcal{F}$, $\mathcal{A}$, I and G. Grounding replaces the variable terms in the parameter lists of the propositions, action schema, goal specifications with constant/non-constant objects.

We defined a domain model for the cyber-security domain using PDDL (Planning Domain Definition Language) a generalization of STRIPS, shown in Figure 4.2. In total this domain model has 45 actions, 47 types, 70 constants and 55 predicates. This STRIPS domain model captures the home computer user cyber-security scenarios simulated in the Psychorithm sandbox environment described in Chapter 3.

Definition 4. *An observation trace O is a sequence of observed actions $O = \{o_1, o_2, ..., o_n\}$ where $o_i \in \mathcal{A}$ and $i = 1, 2, \ldots, n$ (n = length of observation trace) and states resulting from execution of actions $o_i \in O$ are known.*

Traces may contain extraneous or missing actions. An *extraneous action* is an observation $o \in O$ such that the state resulting from executing o is never added to the state (i.e., set of state

```
:(define (domain attackgraph-typed)
      (:requirements :strips :adl :equality :typing
       :disjunctive-preconditions)
      (:types unknown software misc - object)
       account certificate email-ID file key site user attacker
vulnerability-type password direct-message device database parameter
permission process username response request - misc
       browser editor webmailer plug-in site-creating-software
server-connecting-software server anti-virus OS desktop-app server-app
malicious-software - software
       phishing-site server-site normal-site - site
       phishing-site-email phishing-site-twitter - phishing-site
       attacker-remote attacker-local - attacker)

      (:constants
      user1-email-account user1-twitter-account - account
      pc-doctor security-shield - anti-virus
      browser-firefox - browser
      phishing-direct-message - direct-message
      email-with-malicious-attachment - email-ID
      squirrelmail - webmailer
      user1-twitter-password - password ... )

      (:predicates
      (information-available ?aUser - User ?anAccount - account)
      (user-has-email-account ?anAccount - account)
      (direct-message-received ?dmsg - direct-message)
      (anti-virus-software-selected ?av - anti-virus)
      (information-leakage ?anAccount - account ... )

      (:action user-opens-attachment-through-webmail
      :parameters (?U - user ?aBrowser - browser ?WM - webmailer
      ?aSite - site ?anAccount - account ?Msg - email-ID ?F - file )
      :precondition (and (use-software ?aBrowser) (running ?aBrowser)
      (user-visits-site ?U ?aSite) (= ?aSite webmail-site)
      (user-has-email-account ?anAccount)
      (= ?anAccount user1-email-account) (msg-opened ?Msg)
      (mail-attachment ?Msg ?F) (logged-onto-system ?U))
      :effect (opened ?F)))
      ......
```

Figure 4.2: The cyber-security PDDL domain model snippet

variable predicates that are true) by any of the actions $a_i \in \pi_U$. A *missing observation* with respect to π_U occurs when the state resulting from executing o is added to the global state by an of the actions $a_i \in \pi_U$, and $o \notin O$.

When we are looking for possible paths leading to an undesirable state, we search for a **undesirable plan** $\pi_U = \{a_1, \ldots, a_k\}$ of length k that entails one or more undesirable states $U \subseteq \mathcal{F}$ and $a_i \in A$. There may be more than one undesirable plan, in which case we consider a set of undesirable plans Π_U. In the example in Figure 4.1, the domain has only one undesirable state $U = \{U_1\}$.

Definition 5. *An intervention problem $T = \langle \mathcal{D}, O, I, U \rangle$ consists of a planning domain D, a sequence of observed actions $O \subseteq \mathcal{A}$, an initial state $I \subseteq \mathcal{F}$, a set of undesirable states $U \subseteq \mathcal{F}$.*

The solution to the intervention problem is a binary decision for each action in the observation trace, $o \in O; o \rightarrow \{Yes, No\}$ indicating whether it was identified as a critical trigger action (Yes) or not (No).

4.2.2 Identifying Critical Trigger Actions

A **critical trigger action** c_i is a action at $o_i \in O, i \leq |O|$ that maximizes the metric function $V(c_i)$, which we define now.

We recognize two forms of candidate trigger actions in the observation trace: strong and weak. A strong trigger can actively contribute to causing the state (e.g., as an effect or by satisfying a required precondition) or allow another actor to cause it. A trigger may also weakly contribute to the causation of the undesirable state(s) and may not be worth intervening to prevent. Consequently, we add a layer to the intervention problem to support application specific ranking about which actions are most "critical", i.e., helpful in preventing the undesirable state(s). The key addition is the Undesirable Consequence Recognition function that allows a tunable combination of metrics.

Our focus is on identifying the intervention point as early as *helpful* when a user may be triggering an undesirable state. The concept of helpful is similar to the value of alerts described in (Wilkins, Lee, & Berry, 2003) which recognized that it is important to be judicious in interrupting a user in scenarios where humans and machine agents interact with each other: “don’t ignore but don’t annoy”. To this end, we start by defining a function composed of three metrics to quantify the notion of helpfulness: **Certainty**, **Timeliness** and **Desirability**. These metrics are calculated based on a set Π_U.

When producing the set of alternative undesirable plans Π_U, as representative of a set of possible trajectories toward an undesirable goal, it would be best to consider alternatives that are non-obvious and diverse from each other rather than just the optimal plan. This is especially important in long-running processes with complex traces or when a sub optimal plan contains a longer, but perhaps more easy to trigger, path to the undesirable state.

Certainty measures the likelihood of action a occurring in Π_U.

$$Certainty(a|\Pi_U) = \frac{|\pi_U \in \Pi_U \text{ in which } a \text{ occurs}|}{|\Pi_U|} \tag{4.1}$$

Actions occurring frequently in Π_U indicate the importance of that action toward causing U. For example, if an action a occurs in all plans Π_U, the Certainty metric will assign a high value to a, giving it a higher probability of being selected as a critical trigger compared to a less frequent action.

Timeliness requires knowing what actions could yet be observed, which might effect the causation of the undesirable state. For this study, Timeliness is measured by the maximum normalized steps remaining in Π_U in which the action a occurs. Timeliness quantifies how soon an observation will trigger the undesirable state. Therefore, we compute the maximum in discrete time as opposed to the minimum because it indicate that an undesirable state is developing but not imminent and allows time to recover. Let n be the remaining number of steps in $\pi_U \in \Pi_U$ after some action a. Then,

$$Timeliness(a|\Pi_U) = \max_{\pi_U \in \Pi_U} \left(\frac{\max(n)}{|\pi_U|} \right) \tag{4.2}$$

Desirability measures the effect of an action on user’s goals, which is ignored in the other two metrics. In the Intervention by Undesirable Consequence Recognition problem the observer is only aware of the undesirable state the user wants to avoid. Although the user's goal is hidden to the observer, the intervention decision must also consider how the candidate trigger actions contribute toward the user's goal to allow some freedom for the user. In Chapter 3, we showed that in the cyber-security domain, most actions users execute to accomplish a goal are harmless actions while only a few of them put the user at risk. We capture this property with the Desirability metric and separate common harmless actions (e.g., user opening web browser) from avoidable ones and connects the observations to knowledge of the user's goals. If an action appears often in the candidate trigger actions then that action is a common, harmless action and more desirable. In contrast, actions that appear less often in the candidate trigger actions are comparatively more undesirable. The observer must consider the undesirable actions with higher priority. Consequently we downgrade the contribution of the desirable actions to criticality by defining the Desirability as a negative metric.

$$Desirability(a|\Pi_U) = -\left(\frac{|\text{appearance of } a \ \text{ in } \Pi_U|}{\sum_{i=1}^{|\Pi_U|} |\pi_i|}\right) \tag{4.3}$$

The Undesirable Consequence Recognition Function

Together, these metrics define the function $V(a)$ for candidate trigger action a for a weighting provided by α, where $\alpha = [\alpha_1, \ldots, \alpha_m]$, α_i is in the range $[0, 1]$, m is the number of metrics and $\sum_1^m \alpha_i = 1$:

$$V(a) = (\alpha_1 \times Certainty(a|\Pi_U)) + (\alpha_2 \times Timeliness(a|\Pi_U)) + (\alpha_3 \times Desirability(a|\Pi_U)) \tag{4.4}$$

Table 4.2, shows how the critical trigger action is identified using the proposed Undesirable Consequence Recognition function for the example in Figure 4.1 given the observation sequence of actions $O = \{a_1, a_2, a_3\}$. In state S_3, assuming equal weighting of metrics, the algorithm identifies a_{12} to be the action that maximizes the Undesirable Consequence Recognition function, and returns it as a possible point of intervention. Table 4.2 also shows how this decision is affected by the choice of α. If only the Certainty metric is used ($\alpha = [1, 0, 0]$) action f gets selected as the in-

Table 4.2: Certainty (C), Timeliness (T), Desirability (D) computations for each candidate trigger action for the example in Figure 4.1. $V(a)$ is the value returned by the Undesirable Consequence Recognition Function assuming equal weighting ($\alpha = [0.33, 0.33, 0.33]$), only C ($\alpha = [1, 0, 0]$), only T ($\alpha = [0, 1, 0]$) and only D ($\alpha = [0, 0, 1]$) for each candidate trigger action.

	C	T	D	α = [0.33,0.33,0.33]	α = [1,0,0]	α = [0,1,0]	α = [0,0,1]
				$V(a)$	$V(a)$	$V(a)$	$V(a)$
a_5	2/10=0.2	max(3/3,4/4)=1.0	-2/39=0.05	0.38	0.2	**1.0**	-0.05
a_4	4/10=0.4	max(2/2,3/3,5/5,6/6)=1.0	-4/39=0.1	0.43	0.4	**1.0**	-0.1
a_{11}	4/10=0.4	max(3/5,4/6,3/5,4/6)=0.6	-4/39=0.1	0.30	0.4	0.6	-0.1
a_{12}	6/10=0.6	max(2/5,3/6,2/2,3/3,5/5,6/6)=1.0	-6/39=0.15	**0.48**	**0.6**	**1.0**	-0.15
a_8	2/10=0.2	max(2/3,3/4)=0.75	-2/39=0.05	0.30	0.2	0.75	-0.05
a_{10}	1/10=0.1	max(1/2)=0.5	-1/39=0.03	0.19	0.1	0.5	**-0.03**
a_9	3/10=0.3	max(2/3,4/5,5/6)=0.83	-3/39=0.08	0.35	0.3	0.83	-0.08
a_{15}	4/10=0.4	max(2/6,2/3,4/5,5/6)=0.83	-6/39=0.15	0.36	0.4	0.83	-0.15
a_{13}	3/10=0.3	max(1/5,1/2,1/5)=0.5	-3/39=0.08	0.24	0.3	0.5	-0.08
a_6	1/10=0.1	max(2/4)=0.5	-1/39=0.03	0.19	0.1	0.5	**-0.03**
a_7	1/10=0.1	max(1/3)=0.33	-1/39=0.03	0.13	0.1	0.33	**-0.03**
a_{16}	4/10=0.4	max(1/3,1/6,1/3,1/6)=0.33	-4/39=0.1	0.21	0.4	0.33	-0.1
a_{14}	1/10=0.1	max(1/4)=0.25	-1/39=0.03	0.11	0.1	0.25	**-0.03**

tervention point. Using only the Timeliness metric ($\alpha = [0, 1, 0]$) yields three possible intervention points: a_{12}, a_5 and a_4. Finally, using only the Desirability yields four possible intervention points: a_{10}, a_6, a_7 and a_{14}.

Intervention is different from the typical plan recognition problem because the observer assumes that the user pursues some desirable goal but wants to also avoid any undesirable state. Furthermore, the observer has no knowledge about the user's desirable goal and knows only the set of undesirable states the user should avoid. Our model views the user's attempt to achieve a desirable goal as a planning process. Plan recognition approaches that use plan library representation do not apply for the Intervention problem because they are focused on matching observed actions to prior plans to determine what goals the user is trying to achieve. The problems with that are: (1) the observations may contain extraneous actions and (2) the plans that lead to the undesirable states may not encompass the user's goal (the user does not want to trigger the vulnerability, but rather to do something useful with his computer). Therefore, the objective is to identify intervention points based on their potential to cause the undesirable state, while making sure that the actions are identified in an *opportune time*.

Our intervention model borrows ideas from the generative plan recognition approaches and analyses the Planning Graph and operates incrementally. This is similar to prior work by Sun et al. (2007) and Hong (2001). Following work by Geib and Goldman (2009), we adopt a model focused on execution: what could be done next. In contrast to the works of Ramirez and Geffner (2009, 2010) that assumed a fully observable state space, our approach takes into consideration the inherent unreliable nature of observations (missing and irrelevant actions) towards identifying critical trigger actions. Our solution employs PDDL and the Mosaic planner (Roberts, Howe, & Ray, 2014) to sample possible plans from the current state. For each observation, the algorithm outputs whether or not it is a critical trigger action.

4.2.3 Undesirable State Recognition Algorithm

Figure 4.3 shows the seven-step decision cycle of the algorithm. As defined in the problem statement, the process takes as input: a PDDL domain, an initial state, a set of undesirable states and the metric weights for the Undesirable Consequence Recognition Function.

Step 1: We assume the full observation trace is known in advance, with actions made available one at a time to identify the intervention point. However, the algorithm does not utilize information that can be gleaned from the full trace to yield the decision. Input to the algorithm in this step can also be replaced with application specific sensors.

Step 2: For each possible undesirable state U, generate PDDL problem instances. In the first cycle, the initial state is from the input; in subsequent cycles, the state is updated in step 7. To extract applicable states for the next cycle, we chose to search forward the planning graph starting from initial state. This is because the unobservable actions require us to update state during search to accommodate those changes that are presupposed by the actions being able to execute.

Step 3: We opted to apply a diverse planner. A recent study examined the impact of diversity metrics in classical planning (Roberts et al., 2014), and a contribution from that work, the Mosaic planner, produces a set of diverse plans in a single run. A key finding by the authors of Mosaic was that existing diverse planners frequently produce plan sets containing actions not leading to the goal, which increased diversity but lacked justification for many applications. Mosiac returns Π_U for U.

Mosaic is built on top of the LAMA planner (Richter & Westphal, 2010) with some patches applied to its parser from the current Fast Downward repository[1]. It included three extensions to improve the plan set and produce diverse plans faster: the use of a tabu mechanism to guide search away from already known solutions, the use of multiple queues to increase the diversity of solutions explored, and the use of parsimony to bias the search toward goal-oriented solutions. From the family of variants, iterated Tabu A* (ITA) and Multi-queue A* (MQA), we use MQA-TS to compute up to 10 possible plans, based both on the recommendation of the authors and because it showed the most promise in prior work.

Step 4: Extract unique actions in Π_U as candidate trigger actions. Actions include the action name and the instantiated parameter values.

Step 5: Compute the Certainty, Timeliness and Desirability for each candidate trigger action by iterating over the plans in Π_U, as defined in Function 4.4.

Step 6: Rank candidate actions by V. Flag the highest ranked actions as critical triggers corresponding to the current observation. Step 6 also evaluates the accuracy of the identified critical trigger by comparing against the observation. We use a ground-truth **u**ndesirable **p**lan (called UP) that achieves the undesirable state and assume that all actions in the ground-truth undesirable plan are critical triggers. If the algorithm returns the current observation as the critical trigger, then the decision is correct if the current observation is an action in the ground-truth undesirable plan (i.e., true-positive). Alternatively, if the current observation is not the critical trigger and it was also missing in the ground-truth undesirable plan (i.e., observation was an extraneous action and a true-negative), then that decision is also labeled as correct. All other cases are incorrect.

Step 7: The cycle begins over by merging the effects of the observed action as defined in the PDDL domain with the current state. Because we allow for partial observability, the observed action may appear to be inapplicable because actions that satisfied its preconditions were not observed and so captured by the current state. Our approach addresses this inconsistency in state by finding a plan that modifies current state to the state after adding the effects of the observed action.

[1]See `http://www.fast-downward.org/`

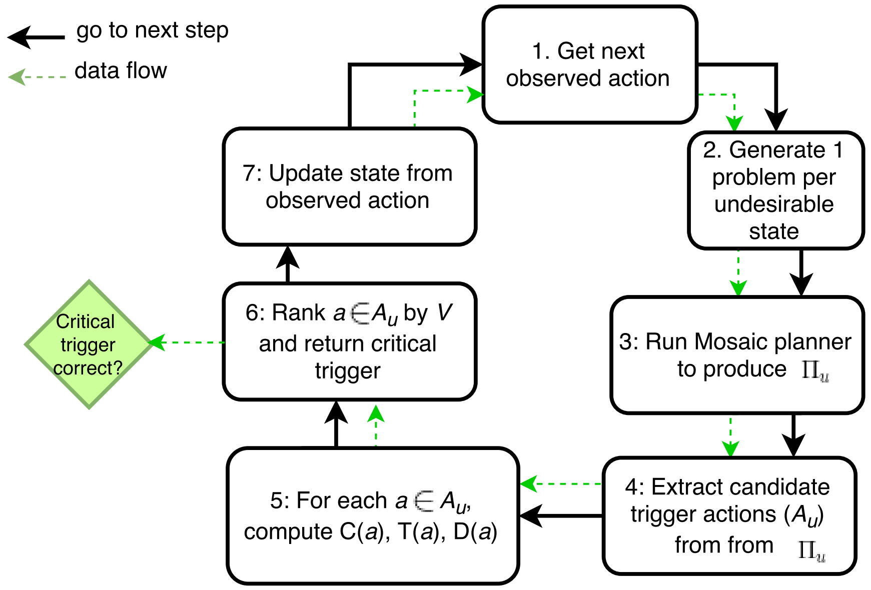

go to next step
data flow
1. Get next observed action
2. Generate 1 problem per undesirable state
3: Run Mosaic planner to produce Π_u
4: Extract candidate trigger actions (A_u) from from Π_u
5: For each $a \in A_u$, compute C(a), T(a), D(a)
6: Rank $a \in A_u$ by V and return critical trigger
7: Update state from observed action
Critical trigger correct?

Figure 4.3: Seven-step process for determining whether an observed action is a critical trigger action.

Then, each action in that plan are executed (i.e., adding add effects, removing delete effects) to update the current state.

4.3 Evaluating the Undesirable Consequence Recognition Function

We examine which factors impact the performance of detecting critical trigger actions using our metric function V. The *independent variables* of the study are summarized in Table 4.3, and include the weights for V, the problems, the percentage of extraneous actions, and the percentage of actions "removed" to emulated partial observability.

The metric weights vary between a single metric, pairs of metrics and all three metrics with equal weights. We decided on these metric weight settings in order to identify which metrics (or their combinations) are sensitive to partial observability and extraneous actions separately.

We begin with results from a user study on cyber-security that motivated our work. This domain will be referred to as the `PAG` in our evaluation. For traces from the user study, we examine the impact of the metric weights (MW) used for in the Undesirable Consequence Recognition function.

Table 4.3: Independent variables for evaluating impact of algorithm parameters and sensitivity to noisy data

Variable	Settings
Metric Weights (MW) (C,T,D)	(1,0,0), (0,1,0), (0,0,1), (0.33,0.33,0.33), (0.5,0.5,0), (0.5,0,0.5), (0,0.5,0.5)
Planning Domains (PD)	PAG, blocks-words, navigator, ipc-grid+, logistics
Partial observability (PO)	25%, 50%, 75%, 100%
Extraneous Actions (EA)	0%, 50%, 75%, 100%

Noisy sensing can lead to extraneous actions or missed observations, which complicates intervention. So we generalize these results to a set of benchmark domains from (Ramírez & Geffner, 2009), where greater experimental control allows us to evaluate the impact of extraneous actions (EA) and partial observability (PO). Thus, we expect some degradation in performance as the noise increases and observability decreases; the question is by how much?

The *dependent variables* of the study include accuracy and computational overhead as measured in CPU time. For the cybersecurity domain, accuracy is defined as the percent true-positives only. For the benchmarks, we report accuracy in terms of percent true-positives and true-negatives. Computation time is included because ultimately we envision this being one component of a user supporting agent, which will require fast response. CPU time includes the time required to process each observation (one cycle of the process depicted in Figure 4.3) within a trial; this includes both generating the alternate plans and ranking the actions.

For each undesirable state, we use a plan generated by an automated planner as the ground-truth undesirable plan. This is a fair assumption because the trace generation algorithm produces traces leading to the undesirable state (with varying levels of noise and observability). We measure accuracy with two evaluation metrics:

- Success rate in ignoring extraneous actions (Ignored EA)
- Success rate in flagging undesirable actions that appear in the ground truth undesirable plan (Flagged UP)

Ignored EA for an observation trace is computed as the count of instances where the observation was an extraneous action and it was not flagged as a critical trigger (count EA). It is given by the equation:

$$\text{Ignored EA\%} = \frac{\text{count EA}}{\text{number of extraneous actions in trace}} \times 100 \tag{4.5}$$

Flagged UP for an observation trace is computed as the count of instances where the observation was an action from UP and the function V actually selected it as the critical trigger action. It is given by the equation:

$$\text{Flagged UP\%} = \frac{\text{count UP}}{\text{number of UP actions in trace}} \times 100 \tag{4.6}$$

4.3.1 Cyber-security Domain (PAG)

In Chapter 3, we discussed the findings of an human subject experiment where users performed routine computer tasks in a sandboxed simulation environment. The objective of this study was to determine how non-expert users behave when presented with questionable computer security situations and to compare their actual observed behavior to their self-reports obtained via pre- and post-hoc surveys. Participants were presented with a desktop which includes common applications such as emailing, Web-browsing, social networking etc. They were also provided with written instructions on how to perform normal home computer user activities such as reading/sending email, installing software etc. While the normal computer activities were being performed, different events were simulated, without the subject's knowledge that can trigger threats and vulnerabilities of interest. These events included enticing user to disclose sensitive information (login names, passwords) to phishing sites over email and social media communications, responding to pop-ups asking the user to download a software, and reacting to malicious activities detected in the computer by an anti-virus program. Sixty-three human subjects participated in the study; their actions were collected into observation traces and will be used in the evaluation of our undesirable consequences recognition algorithm.

We constructed a PDDL domain for the Personalized Attack Graph (`PAG`) to investigate what actions subjects might take amid the aforementioned threat scenarios. We considered four undesirable states for this domain, resulting in four problem definitions. Figure 4.4 shows an actual

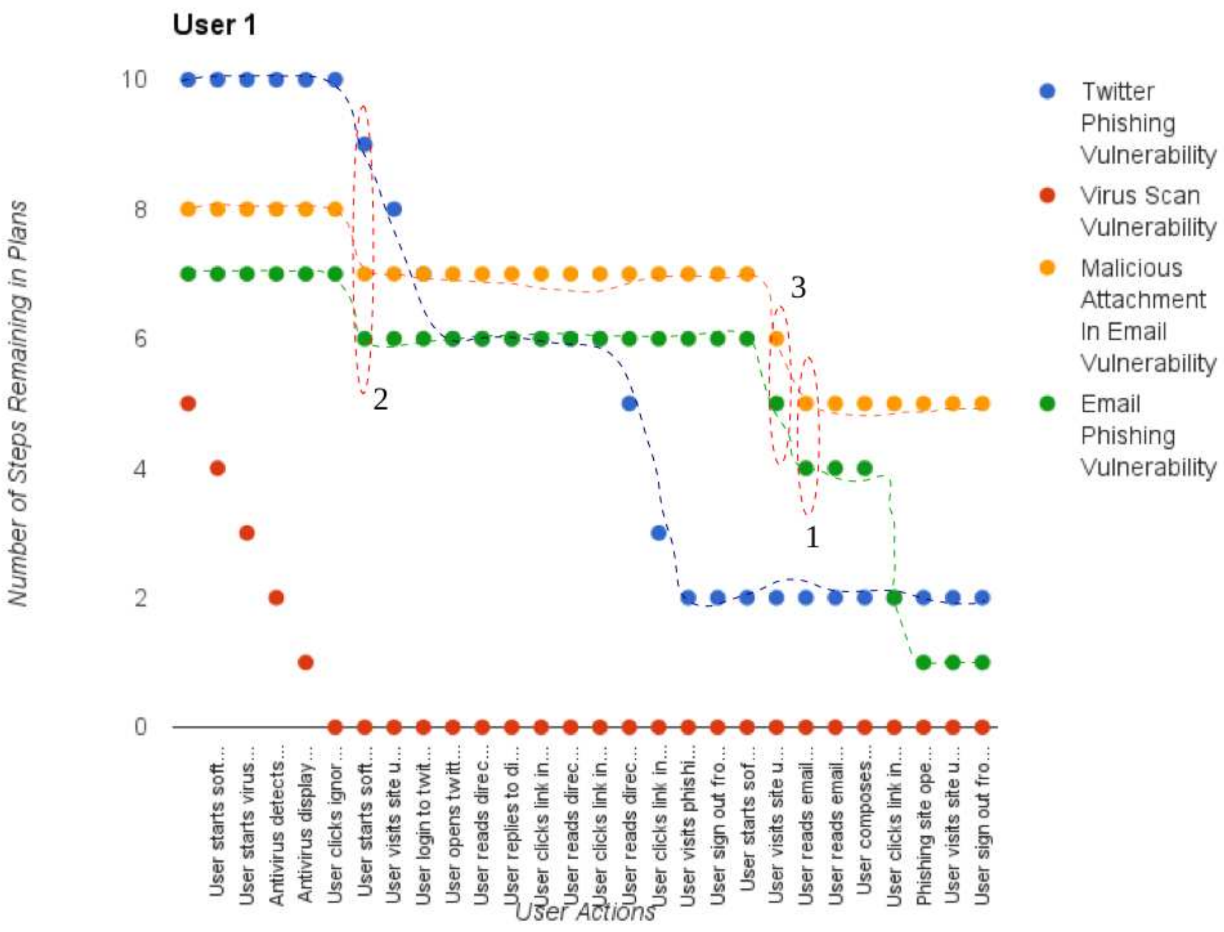


Figure 4.4: Observation trace from a human subject annotated with relation to undesirable plans (colored connected dots) and candidate trigger actions (in dotted ovals)

observation trace indicating how the subject's observed actions (x axis) contributed to triggering the four undesirable states (the plans to do so are displayed as a sequence in each color). The y-axis shows how many steps in each undesirable plan (UP) have yet to be executed. When the dots remain horizontal, the actions are not in the undesirable plan; they are extraneous. For this trace, although the subject came close to triggering all four, only one actually happened (red color line - virus scan vulnerability).

We expected accuracy to be low because user logs offer unique challenges to the algorithm. Human users do not typically perform tasks in an ordered sequence. For example, some logs indicated that the subject was performing the same task repetitively, users had skipped a task to come back to it after an interval and more interestingly, some users were actively trying to trigger the vulnerabilities. These scenarios make it difficult to model a consistent state in a planning environment and affect candidate trigger actions being selected for the critical trigger ranking algorithm. To assess algorithm performance in a more controlled environment, we turned to benchmarks.

PAG Results

Since activity logs captured during the experiment could not be properly controlled for extraneous and missing actions, we limit the assessment to accuracy and the effect of the metric weightings on Flagged UP accuracy and CPU usage.

The subjects were only provided with instructions to perform normal computer tasks. No restrictions were imposed on how they should interact with desktop environment (e.g., undo/redo tasks, exploring application features). Therefore, subject's goals while performing the tasks were unknown. Logs generated by two subjects were discarded because they did not complete the experiment. Considering 61 traces used in this evaluation, mean trace length was 39 (SD=21.13). The longest trace contained 120 observations, while the shortest was 10.

Mean Flagged UP accuracy was 59.53% (SD=30.79) across all metric weight assignment classes, but the weights exerted a significant influence on accuracy (F=40866, $p \ll 0$). The highest accuracy (mean=95.59%,SD=2.13) occurred for two configurations of metric weight assignments: one that equally weighted the Certainty, Desirability and Timeliness metrics as well as the one that equally weighted Certainty and Timeliness, ignoring the Desirability metric. We find that, in the cyber-security domain, the Undesirable Consequence Recognition algorithm relies on the Certainty and Timeliness metrics the most. Mean CPU time per cycle of the algorithm was 1.7 seconds (SD=0.45), suggesting that this algorithm could easily be integrated into an agent that mitigates cyber-security issues.

4.3.2 Benchmark Domains

To generalize the results, we examine four domains from (Ramírez & Geffner, 2009). `Block-Words` constructs one of 20 English words using 8 distinct letters. `Grid-Navigation` forms paths through a connected graph to specific locations (goals). `IPC-grid+` is a grid navigation with keys added at a restricted set of locations. `Logistics` moves packages between locations. To keep the scale similar to the user study, we randomly selected four problems from each domain distribution.

Trace Generation

We choose four undesirable states for each benchmark domain. To construct problem factors, partial observability and extraneous actions, we construct four planning problems with the undesirable states set as the goals. For each problem, we generate noisy traces by incrementally building the plan starting from the initial state and interleaving it with actions from a set of pre-computed extraneous actions when the current state meets the preconditions of the extraneous action. Thus, the trace generation algorithm consists of two stages (1) computing the set of extraneous actions (EA) and (2) interleaving these extraneous actions in to the trace. To construct partial observability (PO), actions from the original ground truth undesirable plan are randomly removed from the interleaved trace.

We use the Metric-FF planner (Hoffmann, 2003) to generate extraneous actions because the planner's implementation offers configuration options to extract the relaxed planning graph and add/delete effects of actions. The challenge in generating extraneous actions is to ensure that they (1) do not introduce new facts that interfere with the progression of the ground-truth undesirable plan or (2) does not delete the progress already made by the ground truth undesirable plan.

First, we incrementally build the set of extraneous actions starting from selecting actions that can be executed immediately after the goal state has been reached. By adding new actions, object types, and predicates to the domain model, the size of the resulting extraneous action set can be increased to handle longer plans and create lengthy traces. The advantage of forward-expanding the set of extraneous actions starting at the goal state is that at this stage, the only condition we need to check is whether or not the chosen action deletes the goal state or not. We add the actions that do not delete the goal state predicates as extraneous actions. It is also possible to select extraneous actions from the applicable actions in the current level of the Relaxed Planning Graph. However, this does not guarantee that these actions will not violate conditions (1) and (2) later in the plan.

In certain occasions selecting extraneous actions this way could lead to traces where the same action is being done and undone repeatedly without making progress through the plan, which is unrealistic and not rational behavior.

We designed a trace generation algorithm that overcomes the aforementioned problems and builds the ground-truth undesirable plan incrementally from the initial state, while interleaving it with extraneous actions. Algorithm 1 describes the process to compute the extraneous action pool for each problem in a domain. We begin by computing the ground-truth undesirable plan UP using

Algorithm 1 Extraneous Action Generation Algorithm

```
1: procedure EXTRANEOUSPOOL($domain$,$undesirableState, init$)
2:     $current \leftarrow UP, EX_g \leftarrow \emptyset, state \leftarrow \emptyset$
3:     $UP \leftarrow Planner(domain, undesirableState, init)$
4:     $state \leftarrow init$
5:     for all $a \in current$ do
6:         if $Pre(a) \in state$ then
7:             $state \leftarrow state \setminus Del(a) \cup Add(a)$
8:     $S_g \leftarrow state$
9:     $RPG \leftarrow RelaxedPlanningGraph(domain, undesirableState, init)$
10:    for all $action \in RPG$ do
11:        if $(action \notin UP)$ & $(Pre(action) \in S_g)$ & $(state \setminus Del(action) \cup Add(action) \cap S_g = \emptyset)$ then
12:            $EX_g \leftarrow action$
13:    return $EX_g$
```

a planner (line 3). Then for the actions in UP we modify the initial state by incorporating the effects of each action. The end of this loop translates the $state$ to the goal state for the problem (lines 5–8). Then we generate the Relaxed Planning Graph for the planning problem with the goal equal to the undesirable state (line 9). We select actions from the Relaxed Planning Graph such that the actions that are not already in UP and the selected action is applicable in the goal state and post-conditions of the selected action do not remove any goal predicates (lines 10–12). These actions form the extraneous action pool EX_g. The set EX_g needs to be extended because we evaluate the Undesirable Consequence Recognition algorithm by varying the percentage of extraneous actions in the observation trace. This is done by modifying S_g with effects of a selected action from EX_g, and repeating steps 4 through 5. The intuition is that once an extraneous action is executed at some state, the resulting state may enable additional extraneous actions. Repeat execution of this process generates a tree structure of extraneous actions with an action from EX_g as root.

For an observation trace, we define the metric percent extraneous actions (EA) as follows:

$$\text{percent extraneous actions (EA)} = \frac{\text{number of extraneous actions in trace}}{\text{number of actions in plan}} \tag{4.7}$$

This metric is used as a proxy for signal-to-noise ratio [0%, 50%, 75%, 100%]. Algorithm 2 describes our trace generation process. Starting from the initial state, actions in the undesirable plan are added to the observation trace sequentially, including extraneous actions as appropriate. This produces a final plan. Partial observability is computed by randomly removing actions from the final plan. The percent partial observability (PO) of the resulting trace is computed as follows:

$$\text{percent partial observability (PO)} = \frac{\text{number of actions in trace after removal}}{\text{number of actions in trace before removal}} \tag{4.8}$$

We create observation traces with [25%, 50%, 75%, 100%] percent partial observability.

Algorithm 2 Observation Trace Generation for Undesirable Consequence Recognition

```
1:  procedure NOISYTRACE($domain$, $undesirableState$, $init$)
2:      $ea \leftarrow 0, trace \leftarrow \emptyset, state \leftarrow init$
3:      $EX_g \leftarrow$ EXTRANEOUSPOOL($domain, undesirableState, init$)
4:      $plan \leftarrow Planner(domain, undesirableState, init)$
5:      while $ea < requiredEA\%$ do
6:          randomly select action $a \in \{plan \cup EX_g\}$
7:          if $Pre(a) \subseteq state$ then
8:              $trace \leftarrow trace \cup a$
9:              $state \leftarrow state \setminus Del(a) \cup Add(a)$
10:             $plan \leftarrow Planner(domain, undesirableState, state)$
11:             update $ea$
12:     Fix PO% in $trace$ by removing actions
13:     return $trace$
```

Benchmark Results

We review the overall accuracy of our algorithm in benchmark domains by evaluating how well it ignores extraneous actions and flags undesirable actions in the ground truth plan.

Ignoring Extraneous Actions

When encountered with extraneous actions in the trace, the algorithm must be able to avoid flagging it as a critical trigger action. We define mean Ignored EA% percentage for a benchmark domain as:

$$\text{mean Ignored EA\%} = \frac{\text{sum of EA\% ignored per trial}}{\text{number of trials with EA\%}>0} \tag{4.9}$$

Table 4.4 shows how mean Ignored EA% varies with the noise level of the observation trace and the metric weight assignments.

Results show that our algorithm consistently ignores extraneous actions for all benchmark domains. This high accuracy rate can be attributed to our process of selecting critical trigger actions. Since candidate trigger actions are extracted from a set of alternative plans leading to the same undesirable state, the likelihood of true extraneous actions to appear in the set of alternative plans is low. This reduces the likelihood of false-positives in the trace.

The metric weight combinations significantly influence Ignored EA% ($p < 0.05$ for the `blo`, `ipc` and `log` domains). Post-hoc analysis using TukeyHSD at $\alpha = 0.05$ shows that across domains, the metric weight combinations that do not consider the Timeliness metric better perform at ignoring extraneous actions than combinations that include the Timeliness. This is because the Timeliness metric is not sensitive to extraneous actions in the trace. As a result, the Undesirable Consequence Recognition function can not sufficiently differentiate between extraneous actions and actions in ground truth plan, leading to false positives. The Timeliness metric captures progress yet to be made from current state to the undesirable state in terms of remaining steps. Extraneous actions do not contribute to the progress toward the undesirable state. Therefore, when extraneous actions are encountered in the trace, the Timeliness metric does not sufficiently demote the extraneous action in the candidate trigger actions set. This leads to false positives. In contrast, the Certainty and Desirability metrics look for occurrences of an action in undesirable plans. As

Table 4.4: Mean Ignored EA% (and standard deviation) for benchmark domains for levels of EA% in observation trace and metric weight assignment classes. The left three columns combine mean Ignored EA% for MW assignments across all values of α and breaks mean Ignored EA% down by EA% in trace. The right MW columns breaks down mean Ignored EA% by specific α value for all levels of EA% in observation trace.

	Mean Ignored EA%									
	EA% in trace			MW Assignments (C,T,D)						
Domain	50%	75%	100%	(0,0,1)	(0,0.5,0.5)	(0,1,0)	(0.3,0.3,0.3)	(0.5,0,0.5)	(0.5,0.5,0)	(1,0,0)
`blo`	95.2 (8.0)	97.6 (3.6)	97.3 (3.2)	99.0 (1.6)	94.7 (6.7)	95.2 (6.2)	95.4 (7.0)	98.6 (1.7)	95.4 (7.0)	98.6 (1.7)
`ipc`	86.4 (14.4)	89.9 (11.6)	89.9 (10.7)	95.6 (3.7)	83.2 (13.0)	81.3 (12.9)	83.9 (15.1)	95.9 (3.4)	85.1 (14.8)	95.9 (3.4)
`log`	97.9 (4.4)	96.7 (5.7)	96.1 (5.4)	98.4 (2.8)	94.4 (7.2)	96.1 (3.6)	96.2 (6.9)	98.6 (2.6)	96.2 (6.9)	98.5 (2.6)
`nav`	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)	100 (0)

extraneous actions do not occur in undesirable plans, both metrics are capable of filtering extraneous actions from the candidate trigger action set by minimizing the metric function value and preventing them from being selected as the critical trigger.

Flagging Undesirable Actions

Flagged UP% captures how well the algorithm flags an action in the observation trace as critical triggers given that the action appears in an undesirable plan treated as the ground truth. We define mean Flagged UP% percentage for a benchmark domain as:

$$\text{mean Flagged UP\%} = \frac{\text{sum of Flagged UP\% per trial}}{\text{number of trials}} \tag{4.10}$$

We first look at Flagged UP% in traces with 0% noise and full observability (i.e., best-case scenario) to establish an upper bound to the metric.

Table 4.5 shows a very large range (Max-Min) for Flagged UP%. This indicates that although the only variable for this sample is the metric weight category, Flagged UP% is also sensitive to other external factors that have not been accounted for in our proposed Undesirable Consequence Recognition function. The challenge lies in identifying these external factors and determining their relationship so that the Undesirable Consequence Recognition function can be tuned to improve accuracy. This will be the main focus of our future work.

Table 4.5: Flagged UP% for traces with 0% EA and 100% PO for `Block-words` (`bw`), `IPC-grid+` (`ipc`), `Logistics` (`log`), and `Grid-Navigation` (`nav`) domains. For each domain, this shows the upper bound for the mean Flagged UP % the algorithm can obtain.

Domain	Flagged UP%			
	Mean	SD	Min	Max
`blo`	37.12	24.83	6.12	77.78
`ipc`	43.27	30.55	0.00	90.00
`log`	31.77	17.27	12.82	66.67
`nav`	61.68	40.31	14.10	100.00

Table 4.6 shows that Flagged UP% also increases when observability increases in the trace. Factor analysis using one-way ANOVA for MW shows that this positive effect is significant ($p < 0.05, df = 6$) for all four benchmark domains. High Flagged UP% was reported for MW combinations that consider Timeliness: specifically for the following configurations,

- Certainty, Timeliness and Desirability weighted equally
- Certainty and Timeliness weighted equally
- Desirability and Timeliness weighted equally

Post-hoc analysis using TukeyHSD at $\alpha = 0.05$ shows that this difference is significant. Thus, we conclude that the Timeliness metric can improve the true-positive rate, yielding higher precision for the algorithm. The Timeliness metric is responds well in identifying critical trigger actions when the observation trace contains partial observability because it captures the number of remaining steps until the undesirable state is reached, sampled from a set likely plans. Even when some actions executed by the user are missed, the observer can accurately select the critical trigger actions using the remaining actions in the sampled plans. Accuracy of the Undesirable Consequences Recognition function when using the Certainty and the Desirability metrics decreases as partial observability increases because these two metrics consider occurrences of an action. When observations are missed, the observer's perception of the current state of the world is different from the user's. When the observer uses the *faulty* current state to sample plans, they may contain actions that are otherwise not needed in reference to the ground truth undesirable plan. This results

Table 4.6: Mean Flagged UP% (and standard deviation) for benchmark domains for levels of PO% in observation trace and the metric weight assignment classes. The left four columns combine mean Flagged UP% for MW assignments across all values of α and breaks mean Flagged UP% down by PO% in trace. The right MW columns breaks down mean Flagged UP% by specific α value for all levels of PO% in observation trace.

	Mean Flagged UP%										
	PO% in trace				MW Assignments (C,T,D)						
Domain	25%	50%	75%	100%	(0,0,1)	(0,0.5,0.5)	(0,1,0)	(0.3,0.3,0.3)	(0.5,0,0.5)	(0.5,0.5,0)	(1,0,0)
`blo`	20.1 (21.2)	23.6 (14.4)	31.8 (19.3)	40.2 (26.1)	9.2 (7.6)	24.8 (20.1)	37.0 (17.5)	46.5 (24.9)	19.0 (7.6)	46.8 (24.9)	19.0 (7.6)
`ipc`	6.9 (18.8)	25.6 (20.0)	42.3 (32.0)	50.8 (36.2)	9.7 (5.2)	46.5 (34.7)	47.9 (35.0)	47.5 (35.8)	10.5 (4.9)	47.1 (35.9)	10.6 (4.9)
`log`	18.4 (27.4)	16.4 (20.1)	20.4 (19.5)	31.5 (17.8)	15.2 (10.6)	20.9 (28.2)	24.6 (25.4)	27.2 (28.5)	18.3 (10.1)	27.2 (28.5)	18.3 (10.9)
`nav`	14.8 (19.1)	32.4 (22.6)	45.2 (27.4)	61.7 (39.8)	14.1 (3.4)	56.8 (33.7)	56.8 (33.7)	56.8 (33.7)	14.1 (3.43)	56.8 (33.7)	14.1 (3.4)

Table 4.7: Mean CPU time in seconds for problem factors for each domain. Δ is the difference between the max and min times for each domain.

Domain	EA% in trace				PO% in trace				
	0%	50%	75%	100%	25%	50%	75%	100%	Δ
`blo`	2.3	2.2	2.2	2.2	2.2	2.2	2.3	2.2	1.4
`ipc`	4.9	4.8	4.9	4.9	4.9	4.9	4.9	4.8	0.9
`log`	1.7	1.7	1.6	1.7	1.7	1.7	1.7	1.7	0.5
`nav`	1.3	1.3	1.3	1.3	1.3	1.3	1.3	1.3	0.5

in candidate trigger action set containing *unwanted* actions, and the two metrics that check for occurrences of an action (Desirability and Certainty) fail to adequately promote actions that must be ranked higher.

Computational Overhead

For each benchmark domain, we calculated the average CPU time per cycle for problem factors (EA and PO). As shown in Table 4.7, we found differences due to observation trace factors; two-way ANOVA on EA and PO showed significant main effects and interaction effects (p <0.05) for `ipc`, `log` and `nav` domains.

4.4 Concluding Remarks

We have described a variant of plan recognition that can help identify actions that warrant intervention to help a user avoid undesirable states while interacting with a computer. We introduced the

Undesirable Consequences Recognition function that combine three domain-indent metrics: Certainty, Timeliness and Desirability. Actions that maximize the Undesirable Consequences Recognition function are selected as actions that warrant intervention. We tested our algorithm on both benchmark planning domains and human subject data from a cyber-security experiment. Results from the benchmark domains show that the Certainty and the Desirability metrics perform well in ignoring extraneous actions, while the Timeliness metric and it's combinations with the Certainty and the Desirability perform well in identifying true positives. We found that the Certainty and the Desirability metrics perform well in recognizing undesirable consequences when the observations contain extraneous actions. We also found that the Timeliness metric respond well in recognizing undesirable consequences when the observation traces contain missing actions. However, for the benchmark domains, the low percentage of true-positives in the case where there is full observability and no extraneous actions indicate that the three metrics are not good enough to recognize intervention for actions in a ground-truth undesirable plan. Therefore, metrics must be developed to evaluate the contribution of less frequent actions appearing in the set of sampled plans. For the cyber-security domain, the Undesirable Consequence Recognition Function relies on the Certainty and Timeliness metrics to flag the true-positives compared to the ground truth undesirable plan.

In this work, we studied how the Undesirable Consequence Recognition Function ignores noise in the trace and flags the true positives considering that the metrics weighted with seven weight classes. The results show that the metric weight class significantly influences the accuracy (i.e., ability to ignore noise and flag the true positives) for the benchmark domains. Therefore, it will be useful to study approaches such as Pareto optimization can be used for identifying the metric weight classes that maximizes the Ignored EA and Flagged UP evaluation metrics.

Chapter 5

Balancing Safety and Freedom for the User: Intervention as Planning and Human-aware Intervention

When working in an unfamiliar online environment, it can be helpful to have an observer that can intervene and guide a user toward a desirable outcome while avoiding undesirable outcomes or frustration. The *Intervention Problem* is deciding when to intervene in order to help a user. The Intervention Problem is similar to, but distinct from Plan Recognition because the observer must not only recognize the intended goals of a user but also when to intervene to help the user, but also to only intervene when necessary. We formalize a family of Intervention Problems and show how these problems can be solved using a combination of Plan Recognition methods and classification techniques to decide whether to intervene. The dimensions of the Intervention Problem presented in this chapter are summarized in Table 5.1

We characterize the observer's decision space as an Intervention Graph and construct it using an "Intervention as Classical Planning" approach to generate potential suffixes of partially executed plans. We extract domain-independent features from this graph and extend several Plan Recognition benchmarks to evaluate this approach. For our benchmarks, the learned models dominate three recent Plan Recognition approaches. We then generalize these results to Human-Aware Interven-

Table 5.1: Dimensions of the Unsafe Suffix Recognition Problem

Dimension	**Domain Specific Properties**
Actors in the environment	User, Competitor (optional), Observer
Goals hidden to the observer	User's goal not hidden One or more known undesirable states
Types of observations	The user's and the competitor's (if present) actions
Noise in observations	None
Intervention recovery	Offer helpful hint

tion, where the observer must decide in real time whether to intervene for human users solving a cognitively engaging puzzle. Using a revised feature set more appropriate to human behavior, we produce a learned model to recognize when a human user is about to trigger an undesirable outcome. We perform a human-subject study to evaluate the Human-Aware Intervention. We find that the revised model also dominates existing Plan Recognition algorithms in predicting Human-Aware Intervention.

5.1 Introduction

Even the best plan can go wrong. Dangers arise from failing to execute a plan correctly or as a result of actions executed by a nefarious agent. Consider route planning where a driver is unaware of upcoming road damage or a traffic jam. Or consider cyber-security where a user is unaware of an unsafe hyperlink. In both, plans achieving the desirable goal have similar prefixes to those that result in undesirable outcomes. Suppose an observer watches the actions of a user working in a risky environment where plans may be subverted to reach an undesirable outcome. We study the problem of how the observer decides whether to intervene if the user appears to need help or the user is about to take an action that leads to an undesirable outcome.

We introduce *Plan Intervention* as a new computational problem and relate it to plan recognition. The Plan Intervention problem can be thought of as consisting of two sub-problems: (1) *Intervention Recognition* and (2) *Intervention Recovery*. In the Intervention Recognition phase, the observer needs to make a decision whether or not the user's likely plan will avoid the undesirable state. Thus, it is possible to argue that if the observer implements an existing state-of-the-art plan recognition algorithm (e.g., Plan Recognition as Planning (Ramírez & Geffner, 2009)), then intervention can take place when the likely goal of the recognized plan satisfies the undesirable state.

In fact, using existing plan recognition algorithms to identify when intervention is required has been studied in several work (Pozanco et al., 2018; Shvo & McIlraith, 2020). A key strategy in these solutions is to reduce the pending goal hypotheses set to expedite recognition. In this chapter, we propose a different approach to perform recognition where the observer uses automated planning combined with machine learning to decide whether the user must be interrupted to avoid u. Our long term objective is to develop this intervention model as an assistive teaching technology to gradually guide human users through cognitively engaging tasks. To this end, for the Intervention Recovery phase we want to enable the observer to actively help the human user recover from intervention and continue the task. We leave Intervention Recovery for future work.

We show that the Intervention Problem carries subtleties that the state-of-the-art Plan Recognition Algorithms do not address where it is difficult for the observer to disambiguate between the d and u. We propose two complementary solutions for Plan Intervention: (1) Unsafe Suffix Intervention and (2) Human-aware Intervention. We show that these two complementary solutions dominate state-of-the-art plan recognition algorithms in correctly recognizing when intervention is required.

Typically, the state-of-the-art plan recognition algorithms reconstruct plan hypotheses from observations. (Ramírez & Geffner, 2009, 2010) reconstruct plan hypotheses by *compiling the observations away* and then using an automated planner to find plans that are compatible with the observations. Another approach generates the plan hypotheses by concatenating the observations with a projected plan obtained from an automated planner (Vered & Kaminka, 2017). The costs of the reconstructed plan hypotheses then lead to a probability distribution over likely goals of the user. Existing plan recognition algorithms require that prior probabilities of likely goals be provided as input.

For intervention, providing goal priors is difficult because certain facts about the domain are hidden to the user and unintended goals may be enabled during execution regardless of the priors. Furthermore, human actors may not construct plans the same way as an automated planner. They may make mistakes early on during tasks having a steep learning curve. Partial knowledge about the domain may preclude the human user from knowing the full effects of his actions or he may not be thinking about the effects at all. Therefore, the observer may not always be able to accurately estimate what the users are trying to do.

We study two kinds of Intervention Problems. In *Unsafe Suffix Intervention*, the observer uses automated planners to project the remaining suffixes and extract features that can differentiate between safe and unsafe plans. We evaluate the recognition accuracy of Unsafe Suffix Intervention on benchmark planning problems. In *Human-aware Intervention*, the observer uses the observed partial solution to extract features that can separate safe and unsafe solutions. We evaluate the accuracy of Human-aware Intervention on a new Intervention Planning benchmark called Rush Hour.

The contributions of this chapter are:

- formalizing the online Intervention Problem to determine when to intervene;
- modeling the observer's decision space as an Intervention Graph;
- defining features to assess the criticality of a state using the Intervention Graph and the sampled plans;
- extending existing benchmarks by Ramirez & Geffner ((2009, 2010)) to incorporate Intervention and evaluating our intervention approach for the extended benchmarks;
- introducing a new Plan Intervention benchmark domain called *Rush Hour*. This is a cognitively engaging puzzle solving task where a player moves vehicles arranged on a grid to clear a path for a target vehicle.
- formalizing the Human-aware Intervention Problem for the Rush Hour planning task and designing features to estimate the criticality using behavior features derived from the observed partial plan;
- presenting the results from a human-subjects study where we collected human behavior on Rush Hour;
- extending an existing plan recognition model to the Intervention Problem; and

- training and evaluating the classification models using Rush Hour puzzle solutions collected from a human subject experiment and showing that the approach works well for the Rush Hour problem.

In this chapter, we first define a general form of the Intervention Problems and introduce three variants: (1) Intervention for a Single User, (2) Intervention in the Presence of a Competitor and (3) Human-aware Intervention. Next, we present approaches for Intervention for a Single User and Intervention in the Presence of a Competitor, both of which use the Intervention Graph and the sampled plans to recognize unsafe suffixes. We evaluate these two approaches against the state-of-the-art plan recognition algorithms on using benchmark planning domains in the benchmarks. Next, we present Human-aware Intervention, that uses machine learning to learn properties of the observed partial plans, as opposed to projecting suffixes, to determine when intervention is required. To evaluate Human-aware Intervention, we introduce a new planning domain called Rush Hour and study how human users solve Rush Hour puzzle as a planning task. We evaluate Human-aware Intervention in the Rush Hour domain compared to state-of-the-art plan recognition algorithms. We conclude the chapter with a discussion on why plan recognition falls short in solving Intervention Problems and present the open questions for future research in designing intervention models.

5.2 The Intervention Environment

We model **intervention** in environments where a *user* is trying to achieve a desirable goal(s), denoted d, while avoiding undesirable outcomes, denoted u. Note that in contrast to the Intervention Problem discussed in Chapter 4, the observer has knowledge about d and u and must take into account both these goal states in order to make the intervention decision. Some environments include a *competitor* who may also take actions in the world, but we assume the user is not aware of the competitor's actions, as might happen in cyber-security applications.

We define the *observer* to be the intervening assistant agent. An *observer* receives each action and decides whether to forward the action to the execution platform; this allows the observer to intervene if the action would result in the undesired outcome u. The observer holds a history of previous observations $H = (o_1, o_2, \ldots, o_{i-1})$ that indicate the actions executed by the user or competitor. The user presents the next action as an observation o_i to the observer and the observer must decide "should I intervene?" This decision necessitates projecting what the observer knows and determining whether the user is about to do something unsafe (u) or is moving too far away from a desirable goal (d). We call such a projection a *suffix*. We denote a single *suffix* projection as X and the set of projections as $\mathcal{X}_\Diamond$ because there will usually be many projections.

The intervention models discussed in Chapters 4 and 5 emphasize analyzing the remaining suffixes to decide when to intervene. In Plan Recognition, the observer uses an observation trace O to derive the user's likely plan. O can be either an ordered sequence of actions (Ramírez & Geffner, 2009, 2010) or an ordered sequence of states (Sohrabi, Riabov, & Udrea, 2016). In the first form of intervention we present in this chapter, the observer considers the remaining plans (i.e., suffixes) in safe and unsafe partitions to learn to recognize unsafe suffixes in order to help the user avoid u. In this case, the observer can help the user by only accepting actions into H that will safely advance the user toward d. We call the first intervention model, *Unsafe Suffix Intervention*. In the second form of intervention, the observer learns to recognize that the user is not making progress toward d by analyzing the H when a suffix is not available. In this case, the observer must offer enough help to *guide* the user toward d without giving the solution. The second intervention model is particularly useful when the user is a human and it can be difficult to evaluate progress with heuristics like a normal planning agent. Therefore, we call the second intervention model, *Human-aware Intervention*. Plan Recognition does not define a method for the user to recover when the observer recognizes a plan leading to u. With the proposed intervention models, we address that limitation with two different types of help the observer can offer to the user.

5.3 Intervention Examples

We will present two examples for *Unsafe Suffix Intervention*. The Grid Navigation domain example is used to illustrate intervention with the user and observer. In the grid navigation task

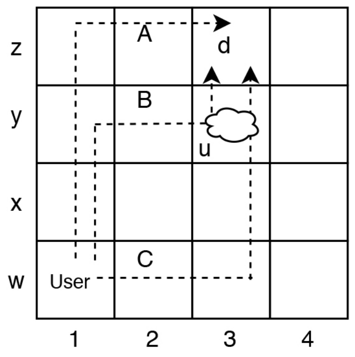


Figure 5.1: Intervention modeled in the Grid Navigation domain

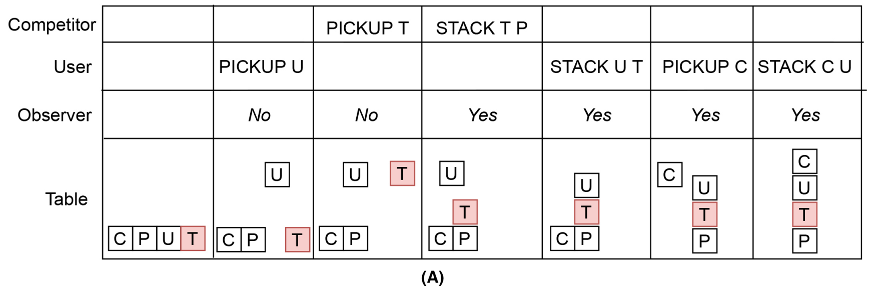
illustrated in Figure 5.1, the user navigates from `W1` to `Z3` by moving vertically and horizontally and `Y3` contains a pit the user can not see: $\mathrm{d} =$ `(AT Z3)` and $\mathrm{u} =$ `(AT Y3)`. Plans corresponding to paths A, B, C are all feasible solutions to the user's planning task. However, plans B and C are unsafe because they satisfy $\{\mathrm{u} \cup \mathrm{d}\}$.

Table 5.2 shows how an observer modeled as an offline Plan Recognition agent will recognize the goals given observations O for the Grid Navigation problem in Figure 5.1. Let us assume that the observer implements the Plan Recognition as Planning algorithm introduced by Ramirez et al. (2010) to disambiguate between u and d. They show that the most likely goal will be the one that minimizes the cost difference $c(\mathrm{g}|O) - c(\mathrm{g}|\overline{O}), \mathrm{g} \in \{\mathrm{d}, \mathrm{u}\}$. For each incrementally revealed O (shown in columns), assuming uniform goal priors, the observer finds the most likely goal that agrees with O. For the cost computation, we assumed that the user is following a satisfising plan to achieve goals. It can be seen that for the Grid Navigation example, the observer can not correctly disambiguate between d and u. More importantly, the final last column satisfies u. At this point, it will be too late for the user to avoid u. Furthermore, because the observer accepts O as is for Plan Recognition, the observer can not guide the user away from u as the observations are already satisfied in the state.

Now let us consider a situation where there may be some additional agent taking actions in the world. For example, in a cyber-security application, a second agent may insert malicious code in a file to gain access to privileged information. More generally, we call this additional agent a competitor, since it is not always the case that they are "attacking" a user. The Blocks Word domain example from Ramirez et al. (2009) is used to illustrate intervention with the user, the competitor

Table 5.2: Observer modeled as a Plan Recognition agent for the Grid Navigation example in Figure 5.1

O	(MOVE W1 X1)	(MOVE W1 X1 MOVE X1 Y1)	(MOVE W1 X1 MOVE X1 Y1 MOVE Y1 Y2)	(MOVE W1 X1 MOVE X1 Y1 MOVE Y1 Y2 MOVE Y2 Y3)
$c(\mathrm{u}\|O) - c(\mathrm{u}\|\overline{O})$	$4 - 4 = 0$	$4 - 4 = 0$	$4 - 4 = 0$	$4 - 4 = 0$
$c(\mathrm{d}\|O) - c(\mathrm{d}\|\overline{O})$	$5 - 5 = 0$	$5 - 5 = 0$	$5 - 5 = 0$	$5 - 5 = 0$
Most likely goal	No decision	No decision	No decision	Fail



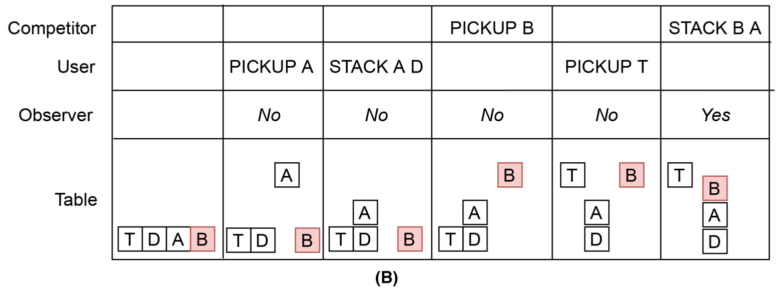


Figure 5.2: (A) User and competitor intervention modeled in the Blocks Words domain, where $\mathrm{u} = $ (CUT) and $\mathrm{d} = $ (CUP). (B) User and competitor intervention where $\mathrm{u} = $ (BAD) and $\mathrm{d} = $ (TAD).
Initially, all four blocks are on the table. An action in the user or competitor row indicates the Intervention Suffix. The *Yes* label in the observer's row indicates that intervention is required. The *No* label indicates that intervention is not required.

and the observer. In the Intervention Problem illustrated in Figure 5.2 (A), the competitor's goal $\mathrm{u} = \{$`(CLEAR C)(ON C U)(ON U T)`$\}$(i.e., CUT) and the user's goal $\mathrm{d} = \{$`(CLEAR C)(ON C U)(ON U P)`$\}$(i.e., CUP). The user can not recognize the block T (shown in red). The competitor only modify the state of block T and executes actions with the block T. As shown in the first column, initially, all four blocks are on the table. Both the user and the competitor incrementally reveal their plans. The user's and the competitor's rows in Figure 5.2 (A) show an example reveal sequence from left to right. The row for the table shows the resulting states after each reveal. In this example, the observer needs to recognize that when the competitor reveals `STACK T P`, it becomes impossible for the user to avoid u. However, the user may continue to reveal actions because he can not recognize the post-condition of `STACK T P`. Therefore, any subsequent reveal must also be recognized as unsafe. The *Yes* labels in the observer's row indicate that intervention is required. Note that in the Blocks Words example, d and u are distinct enough to disambiguate early.

Table 5.3 shows how an observer modeled as an offline Plan Recognition agent will recognize the goals for the Blocks Words problem in Figure 5.2 (A). Similar to the Grid Navigation example, the Plan Recognition agent implements the Plan Recognition as Planning algorithm introduced by Ramirez et al. (2010) to disambiguate between u and d. All other assumptions in the Grid Navigation Plan Recognition task also hold in this problem. It can be seen that in the Blocks Words example the observer disambiguate u and d somewhat better than the Grid Navigation example, because the goals are different enough. The most likely goal identified by the observer aligns with the Yes/No decisions in Figure 5.2 (A). However, there are still "observation reveals" where the observer can not make a clear decision. In the last reveal, the observer correctly identifies u as the goal. However, the observer can not help the user avoid u in the final reveal because O satisfies u by definition of O in Plan Recognition as Planning. With the proposed intervention algorithms, which learn the distinctions between unsafe and safe plans, we hope to improve the intervention recognition accuracy for the observer, while allowing the user some freedom to satisfy d avoiding

Table 5.3: Observer modeled as a Plan Recognition agent for the Blocks Words example in Figure 5.2 (A)

O	(PICKUP U)	(PICKUP U PICKUP T)	(PICKUP U PICKUP T STACK T P)	(PICKUP U PICKUP T STACK T P STACK U T)	(PICKUP U PICKUP T STACK T P STACK U T PICKUP C)	(PICKUP U PICKUP T STACK T P STACK U T PICKUP C STACK C U)
$c(\mathrm{u}\|O) - c(\mathrm{u}\|\overline{O})$	$4-4=0$	$8-4=4$	$8-4=4$	$8-4=4$	$8-4=4$	$8-4=4$
$c(\mathrm{d}\|O) - c(\mathrm{d}\|\overline{O})$	$4-4=0$	$6-4=2$	$10-4=6$	$16-4=12$	$18-4=14$	$20-4=16$
Most likely goal	No decision	d	u	u	u	Fail

u. Furthermore, by accepting actions that only help the user advance toward d safely into the observation history H, our intervention algorithm ensures that the user avoids u.

5.4 Defining Intervention Problems

In this section we outline our main assumptions (Section 5.4.1), discuss the STRIPS planning model (Section 5.4.2), discuss an important notion of direct or indirect actions leading to u (Section 5.4.3), and define the general form of the Intervention Problem as well as highlight the family of problems we study in this paper (Section 5.4.4).

5.4.1 Preliminaries and Assumptions

Because the observer's objective is to help the user safely reach d, an intervention episode (i.e., a sequence of intervention decisions) is defined from the initial state s_0 until d is satisfied. If the competitor is present, the observer decides which actor's presented action to process first, randomly. The actor(s) take turns in presenting actions from their respective domain definitions to the observer until the intervention episode terminates.

We make some simplifying assumptions in this study. **Observability:** The observer has full observability, and knows about states d and u. u is unknown to the user. d is unknown to the competitor. When we say *unknown*, it means that the agent does not actively execute actions to enable the unknown goal. The user cannot recognize the effects of a competitor's actions. **Plans:** The user follows a satisficing plan to reach d, but may reach u unwittingly. There is a satisficing plan to reach $\mathrm{u} \cup \mathrm{d}$ and we assume that it has a common prefix with a plan to reach d. We assume that the user continues to present the observer with actions from his original plan even after the first positive flag and does not re-plan. **Competitor:** When present, the competitor only performs actions using objects hidden to the user; this restriction follows from many security domains where an attacker is a remote entity that sets traps and expects the user to become an unwitting accomplice. The user and the competitor are (bounded) rational agents.

5.4.2 The Intervention Model

We model the users in the intervention environment as STRIPS planning agents (Fikes & Nilsson, 1971). A STRIPS planning domain is a tuple $D = \langle F, A, s_0 \rangle$ where F is the set of fluents, $s_0 \subseteq F$ is the initial state, and A is the set of actions. Each action $a \in A$ is a triple $a = \langle Pre(a), Add(a), Del(a) \rangle$ that consists of preconditions, add and delete effects respectively, where $Pre(a), Add(a), Del(a)$ are all subsets of F. An action a is applicable in a state s (represented by subsets of F) if preconditions of a are true in s; $pre(a) \in s$. If an action a is executed in state s, it results in a new state $s' = (s \backslash del(a) \cup add(a))$, and defined by the state transition function $\gamma(s, a) = s'$. A STRIPS planning problem is a tuple $P = \langle D, G \rangle$, where D is the STRIPS planning domain and $G \subseteq F$ represents the set of goal states. A solution for P is a plan $\pi = \{a_1, \ldots, a_k\}$ of length k that modifies s_0 into G by execution of actions $a_1, \ldots, a_k$. The effect of executing a plan is defined by calling γ recursively: $\gamma(\ldots \gamma(\gamma(s_0, a_1), a_2) \ldots, a_k) = G$.

The Intervention Problem requires domain models that are distinct for the user, observer, and competitor. Let us define the user's domain model as $D_{\text{user}} = (F_{\text{user}}, A_{\text{user}}, s_0)$. The competitor can see the effects of the user but cannot take user actions. We denote the domain model for the competitor as $D_{\text{other}} = (F_{\text{user}} \cup F_{\text{other}}, A_{\text{other}}, s_0)$. Although the observer can see the actions of the user and the competitor, but does not execute actions. This is because we are only focused on the recognition aspect of the Intervention Problem. So the observer's domain model is $D_{\text{observer}} = (F_{\text{user}} \cup F_{\text{other}}, A_{\text{user}} \cup A_{\text{other}}, s_0)$.

5.4.3 The Unsafe Intervention Suffix and Direct/Indirect Contributors

As seen in Section 5.2, when presented with an action, the observer must intervene after analyzing the remaining plans considering u and $\mathrm{u} \cup \mathrm{d}$. The Intervention Suffix analysis allows the observer to identify the observations that help the user avoid u.

Definition 6 (Intervention Suffix)**.** *Let a_k be an action that achieves some goal g from state s_{k-1} (i.e., $\gamma(s_{k-1}, a_k) = g$). An Intervention Suffix $X_g = (a_1, a_2, \ldots, g)$ is a sequence of actions that starts in a_1 and ends at $g \subset \{\mathrm{u}, \mathrm{u} \cup \mathrm{d}\}$.*

Suppose that we want to determine a path to u where the Intervention Suffix is $X_{\mathrm{u}} = (o_i, \ldots, \mathrm{u})$. By replacing g with u, or $\mathrm{u} \cup \mathrm{d}$, an automated planner can be used to generate an Intervention Suffix. We use the set of Intervention Suffixes ($\mathcal{X}_{\Diamond}$), where $X_{\mathrm{u}}, X_{\mathrm{u}\cup\mathrm{d}} \in \mathcal{X}_{\Diamond}$ generated by the Top-K planner (Riabov et al., 2014) to evaluate Unsafe Suffix Intervention in Section 5.6. We refer to a single suffix (i.e., plan) leading to u as π_{u} and a set of such plans as Π_{u}. Similarly, we refer to suffixes leading to d and avoids u as π_{d} and a set of such plans as Π_{d}.

Actions may directly or indirectly contribute to u. The direct and indirect contributors express different degrees of urgency to intervene. A directly contributing action indicates that u is imminent and intervention must happen immediately. An indirectly contributing sequence indicates that u is not imminent, but intervention may still be an appropriate decision. Next, we define directly contributing actions and indirectly contributing sequences.

Definition 7 (Direct Contributor)**.** *A* directly contributing action *a_{crit} occurs in an undesirable plan $\pi_{\mathrm{u}} \in \Pi_{\mathrm{u}}$ and execution of a_{crit} in state s results in a state s' such that $\gamma(s, a_{crit}) = \mathrm{u}$.*

Example of a directly contributing action. In the example illustrated in Figure 5.2 (B), $\mathrm{d} = \{$`(ON T A)(ON A D)`$\}$(i.e., TAD) and $\mathrm{u} = \{$`(ON B A)(ON A D)`$\}$ (i.e., BAD). The user may enable d when he reveals $\{$(STACK USER A D)$\}$, but at the same time this create an opportunity for the competitor to reach u first. However, the observer flags $\{$(STACK COMPETITOR B A)$\}$ for intervention (marked *Yes*) because the post-condition of STACK B A satisfies u. Therefore, STACK COMPETITOR B A is a *directly contributing action*.

Definition 8 (Indirect Contributor)**.** *An* indirectly contributing sequence q_{crit} *is a totally ordered action sequence in an undesirable plan in* Π_{u} *and the first action in* q_{crit} *is equal to the first action in the Intervention Suffix* $x_1 \in X_{\mathrm{u}\cup\mathrm{d}}$. *Executing actions in* q_{crit} *from state* s *results in a state* s' *such that* $\gamma(s, q_{crit}) = \{\mathrm{u} \cup \mathrm{d}\}$.

Example of an indirectly contributing sequence. Figure 5.2 (A) illustrates an *indirectly contributing sequence*. The totally ordered sequence $\{$STACK COMPETITOR T P, STACK USER U T, PICKUP USER C, STACK USER C U$\}$ is an *indirectly contributing sequence* because the actions in the sequence together satisfies $\mathrm{u} \cup \mathrm{d}$. Any Intervention Suffix X_g containing actions from an indirectly contributing sequence must be flagged for intervention.

We next formally define the Unsafe Intervention Suffix X_{unsafe}.

Definition 9 (Unsafe Suffix)**.** *An Intervention Suffix* X *of length* k *is unsafe if there is at least one action* $x_i \in X (1 \leq i \leq |X|)$ *such that* x_i *is a directly contributing action or* x_i *is in a indirectly contributing sequence.*

In the example in Figure 5.2 (B), $X_{\text{unsafe}} = ($PICKUP USER A, STACK USER A D, PICKUP COMPETITOR B, PICK USER UP T, STACK COMPETITOR B A, u $)$ because of the directly contributing action STACK COMPETITOR B A. In the example in Figure 5.2 (A), $X_{\text{unsafe}} = ($PICKUP USER U, PICKUP COMPETITOR T, STACK COMPETITOR T P, STACK COMPETITOR T P, STACK USER U T, PICKUP USER C, STACK USER C U, $\mathrm{u} \cup \mathrm{d}$) because it contains the actions from an indirectly contributing sequence.

5.4.4 The Family of Intervention Problems

We now define a general form of the Intervention Problem. Let $plan(o_i, g)$ be some general method to generate suffixes for π_{d} and π_{u}; in Section 5.5.2 we will show how we can use classical planning.

Definition 10 (Intervention Problem)**.** *Let $\mathcal{I} = (D, \mathrm{d}, \mathrm{u}, H, o_i, \mathcal{X}_\Diamond)$ be a tuple where $D = \langle F, A, s_0 \rangle$ is a planning domain, $\mathrm{d} \subset F$ is a desirable state, $\mathrm{u} \subset F$ is an undesirable state, $H = (o_1, o_2, \ldots, o_{i-1})$ is a history of previously observed actions, o_i is the* presented action *that the user would like to perform, and $\mathcal{X}_\Diamond = \{X_j = plan(o_i, g) | \forall g \in \{\mathrm{u}, \mathrm{u} \cup \mathrm{d}\}\}$ for $j \geq 0$ is a set of suffixes leading to u and $\mathrm{u} \cup \mathrm{d}$. The* Intervention Problem *is a function $intervene(\mathcal{I}) : \mathcal{I} \rightarrow \{No, Yes\}$ that determines for the presented action o_i whether to intervene.*

To decide whether $\mathcal{X}_\Diamond$ contains an unsafe suffix, the observer analyzes suffixes for $plan(o_i, \mathrm{u})$, and $plan(o_i, \mathrm{u} \cup \mathrm{d})$. If the observer finds that $\mathcal{X}_\Diamond$ contains an unsafe suffix then o_i is not accepted into H. Let history $H = (o_1[s_1], o_2[s_2], \ldots, o_{i-1}[s_H])$ be a sequence of previously observed actions, which started from s_0 with the implied resulting states in brackets. The state resulting from applying history to s_0 is $s_H = \gamma(s_0, H)$. If o_i is accepted, then $H' = \{H \cup o_i\}$ and the effect of o_i is represented in state as defined by $\gamma(s_H, o_i)$. A solution to $\mathcal{I}$ is sequence of $\{No, Yes\}$ decisions for each step i of observations. We next explore special cases of the Intervention Problem, namely single-user intervention, competitive intervention, and the most general form of multi-agent intervention.

Intervention for a Single User

When only the user and the observer are present, the user solves the planning problem $P_{\text{user}} = \langle F_{\text{user}}, A_{\text{user}}, s_0, \mathrm{d} \rangle$, and incrementally reveals it to the observer. At each point in the plan solving P_{user}, the observer must analyze $I = (D_{\text{user}}, \mathrm{d}, \mathrm{u}, H, o_i, \mathcal{X}_\Diamond)$, where $\mathcal{X}_\Diamond$ is generated in some sensible way.

Intervention in the presence of a Competitor

If a competitor is present, the user's planning problem P_{user} is the same as before. However, the competitor also solves a planning problem $P_{\text{other}} = \langle F_{\text{user}} \cup F_{\text{other}}, A_{\text{other}}, s_0, \mathrm{u}\rangle$. Note that, the competitor has a limited set of actions in A_{other} to create states that will lead to u and $A_{\text{other}} \cap A_{\text{user}} = \emptyset$. The user's and the competitors solutions to P_{user} and P_{other} are revealed incrementally. Therefore, $H, \mathcal{X}_{\Diamond} \subset \{A_{\text{user}} \cup A_{\text{other}}\}$. To decide whether $X \in \mathcal{X}_{\Diamond}$ is unsafe, the observer analyzes $I = (D_{\text{other}}, \mathrm{d}, \mathrm{u}, H, o_i, \mathcal{X}_{\Diamond})$, where $D_{\text{other}} = \langle F_{\text{user}} \cup F_{\text{other}}, A_{\text{other}} \cup A_{\text{user}}, s_0\rangle$. The observer accepts o_i into H as before.

Human-Aware Intervention

When performing tasks with a steep learning curve (e.g., a puzzle, problem solving), human users may initially make more mistakes or explore different (sub-optimal) solution strategies. Over time, a human may learn to make better choices that result in more efficient plans. Because of the inconsistencies in solution search strategy, we cannot accurately project the goals of the human user. Learning properties about the history H will help the observer recognize when the user is about to make a mistake and use that information to guide the search task on behalf of the user. When humans are solving problems in real time, the criteria for intervention may place more emphasis on the history H than on the suffixes $\mathcal{X}_{\Diamond}$. In Section 5.7, we consider the special case where $\mathcal{X}_{\Diamond} = \emptyset$.

5.5 Recognizing Unsafe Suffixes

We present two solutions for recognizing unsafe suffixes. Recall that in Definition 10, the observer's decision space $\mathcal{X}_{\Diamond}$ is derived such that $\mathcal{X}_{\Diamond} = \{X_j = plan(o_i, g) | \forall g \in \{\mathrm{u}, \mathrm{u} \cup \mathrm{d}\}\}$ for $j \geq 0$. In the first solution, we implement the function $plan(o_i, g)$ as an Intervention Graph (in Section 5.5.1 and Section 5.5.2). In the second solution, we implement $plan(o_i, g)$ by sampling the plan space using an automated planner and use plan distance metrics to make the decision about whether to intervene (Section 5.5.3).

5.5.1 The Intervention Graph

The Intervention Graph models the decision space of the observer for the Intervention Problem $\mathcal{I}$. We can extract several features from the Intervention Graph to derive functions that map the presented observation o_i to intervention decisions. The Intervention Graph captures where u, and $\mathrm{u} \cup \mathrm{d}$ lie in the projected state space from s_H. We can use properties of the graph to evaluate how close the current projection H is to u and $\mathrm{u} \cup \mathrm{d}$ and identify directly and indirectly contributing actions.

The Intervention Graph consists of alternating state and action layers where each state layer consists of predicates that have been made true by the actions in the previous layer. The root node of the tree is s_H. An action layer consists of actions (defined in D_{user} or D_{other}) whose preconditions are satisfied in the state from the previous layer. Algorithm 3 describes the process of building the Intervention Graph. The algorithm takes as input a domain theory D (for D_{user} or D_{other}), s_H and $g = \{\mathrm{u}, \mathrm{u} \cup \mathrm{d}\}$ (lines 1-2). When $H = \emptyset$, the root of the tree is set to s_0. Next, using the domain theory, actions whose preconditions are satisfied at current state are added to the graph (lines 5-6). Each action in level i spawns possible states for level $i + 1$. Line 7 ensures that the actions that immediately inverts the previous action are not added to the graph. For each resulting state a search node is created, with an edge representing the action responsible for the state transition (lines 8-10). The method is executed recursively for each open search node until d and u are added to the graph generates $\mathcal{X}_\Diamond$ for the observer (line 11). To ensure that only realistic plans are explored, we do not add no-op actions to the action layers in the graph. When the user and the competitor present new actions, the root of the graph is changed to reflect the new state s_H and subsequent layers are modified to that effect.

Algorithm 3 Build Intervention Graph

```
Require: $D$, $s_H$, $g$
1: $i = 0$; $s_i \leftarrow s_0$
2: procedure EXPANDGRAPH($D, s_H, g$)
3:     if $s_i \models g$ then return $\langle V, E \rangle$
4:     else
5:         for action $a$ where $Pre(a) \in s_i$ do
6:             $s_{i+1} \leftarrow ((s_i \setminus Del(a)) \cup Add(a))$
7:             if $s_{i+1} \equiv s_i$ then continue
8:             $v \leftarrow$ AddVertex ($s_{i+1}$)
9:             $e \leftarrow$ AddEdge ($s_i, s_{i+1}, a$)
10:            $V \cup \{v\}$ ; $E \cup \{e\}$
11:            ExpandGraph ($D, s_{i+1}, g$)
```

The Intervention Graph is a weighted, single-root, directed acyclic connected graph $IG = \langle V, E \rangle$, where V is the set of vertices denoting possible states the user could be in leading to g, and E is the set of edges representing actions from D_{user} or D_{other} depending on single user intervention or competitive intervention. X_{unsafe} is a path from the root of the IG to $\mathrm{u} \cup \mathrm{d}$ or u. In contrast, a safe suffix X_{safe} is a path from the root of the IG to d and avoids u.

5.5.2 Intervention Graph Features

We extract a set of features from the Intervention Graph that help determine whether to intervene. These features include: Risk, Desirability, Distance to d, Distance to u and Percentage of active undesirable landmarks. We use these features to train a classifier that learns to identify actions in a_{crit} and q_{crit}. Figure 5.3 illustrates a fragment of the Intervention Graph from Figure 5.2 (B) after the user presents the action `PICK-UP A`, which we will use as a running example to discuss feature computation.

Risk (R)

Risk quantifies the probability that the presented action will lead to u. We model the uncertainty the observer has about the next action as a uniform probability distribution across the set

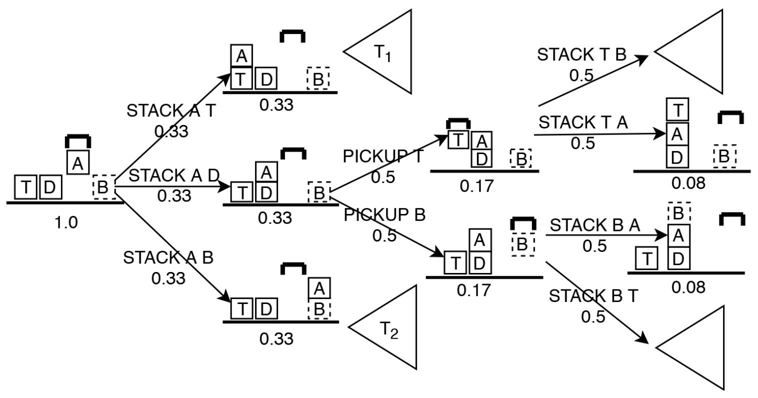


Figure 5.3: Fragment of the decision space after PICKUP A has been proposed for block-words example in Figure 5.2 (B). Numbers under each state and action indicate the probability. Sub trees T_1 and T_2 are not expanded for simplicity.

of applicable actions whose preconditions are satisfied in current state. We define risk R as the posterior probability of reaching u while the user is trying to achieve d. We extract $\mathcal{X}_\Diamond$ from the Intervention Graph by searching breadth-first from the root until vertices containing d is found, including the paths in which the user has been subverted to reach u. By construction, d will always be a leaf. Let $|\mathcal{X}_\Diamond| = n$. The set of unsafe intervention suffixes, $\mathcal{X}_\blacksquare$ is such that $\mathcal{X}_\blacksquare \subseteq \mathcal{X}_\Diamond$ and $|\mathcal{X}_\blacksquare| = m$ and $(m \leq n)$. We compute posterior probability of reaching u for $\mathcal{X}_\blacksquare$, using the chain rule in probability as, $Pr_{\text{unsafe}} = \prod_{i=k}^{1} P(\alpha_i|\alpha_{i-1}\dots\alpha_1)$, and $\alpha_j \in \{A_{\text{user}}\}$ or $\alpha_j \in \{A_{\text{user}} \cup A_{\text{other}}\}$ and k is the length of the suffix until u is reached. Then:

$$R = \begin{cases} \frac{\sum_{i=1}^{m} Pr_{\text{unsafe}_i}}{m} & m > 0 \\ 0 & m = 0 \end{cases}$$

In Figure 5.3, $(n = 6)$ and $(m = 1)$. Since we assumed full observability for the observer, the root of the tree (current state) is assigned the probability of 1.0. Actions that are immediately possible after the current state are each assigned probabilities following a uniform distribution across the branching factor (0.33). Then for each applicable action in the current state, the resulting state gets the probability of $(1.0 \times 0.33 = 0.33)$. Similarly, we apply the chain rule of probability for each following state and action level in the graph until u first appears in the suffix. $R = \frac{0.08}{1} = 0.08$.

Desirability (D)

Desirability measures the effect of the observed action to help the user pursue d safely. Let $\mathcal{X}_\square$ be the set of suffixes that reach d while avoiding u. Then $\mathcal{X}_\square = \mathcal{X}_\Diamond \setminus \mathcal{X}_\blacksquare$. We compute posterior probability of reaching d avoiding u for $\mathcal{X}_\square$, using the chain rule in probability as, $Pr_{safe} = \prod_{i=k}^{1} P(\alpha_i|\alpha_{i-1}\dots\alpha_1)$, and $\alpha_j \in \{A_{\text{user}}\}$ or $\alpha_j \in \{A_{\text{user}} \cup A_{\text{other}}\}$ and k is the length of path. Then:

$$D = \begin{cases} \frac{\sum_{i=1}^{|\mathcal{X}_\square|} Pr_{safe_i}}{|\mathcal{X}_\square|} & |\mathcal{X}_\square| > 0 \\ 0 & |\mathcal{X}_\square| = 0 \end{cases}$$

In Figure 5.3, there are five paths where user achieved d without reaching u (two in subtree T_1, three in the expanded branch). Following the same approach to assign probabilities for states and actions, $D = \frac{(0.08+0.08+0.08+0.04+0.04)}{5} = 0.07$. R and D are based on probabilities indicating the confidence the observer has about the next observation.

Distance to u (δ_{u})

This feature measures the distance to state u from the current state in terms of the number of edges in the paths in $\mathcal{X}_\Diamond$ extracted from the Intervention Graph. We extract $\mathcal{X}_\Diamond$ from the Intervention Graph from the root to any vertex containing d, including the paths in which the user has been subverted to reach u instead. Let $|\mathcal{X}_\Diamond| = n$. The set of suffixes that reach u, $\mathcal{X}_\blacksquare$ is such that $\mathcal{X}_\blacksquare \subseteq \mathcal{X}_\Diamond$ and $|\mathcal{X}_\blacksquare| = m$ and $(m \leq n)$. We count s, the number of the edges (actions) before u is reached for each path in $\mathcal{X}_\blacksquare$ and δ_{u} is defined as the average of these distance values:

$$\delta_{\mathrm{u}} = \begin{cases} \frac{\sum_{i=1}^{m} s_i}{m} & m > 0 \\ -1 & m = 0 \end{cases}$$

In this formula, -1 indicates that the undesirable state is not reachable from the current state. For the example problem illustrated in Figure 5.3, $\delta_{\mathrm{u}} = \frac{3}{1} = 3$.

Distance to d (δ_{d})

This feature measures the distance to d from current state. The path set $\mathcal{X}_{\square}$ contains action sequences that reach d without reaching u. We count t, the number of the edges where d is achieved safely for a path in $\mathcal{X}_{\square}$. Then, δ_{d} is defined as the average of these distances given by the formula:

$$\delta_{\mathrm{d}} = \begin{cases} \frac{\sum_{i=1}^{|\mathcal{X}_{\square}|} t_i}{|\mathcal{X}_{\square}|} & |\mathcal{X}_{\square}| > 0 \\ -1 & |\mathcal{X}_{\square}| = 0 \end{cases}$$

In this formula, -1 indicates that d cannot be reached safely from the current state. For the example problem illustrated in Figure 5.3, $\delta_{\mathrm{d}} = \lceil \frac{3+3+7+7+3}{5} \rceil = 5$.

Active attack landmark percentage ($\mathcal{L}_{ac}$)

This feature captures the criticality of the current state toward contributing to u. We used the algorithm proposed by (Hoffmann et al., 2004) to extract fact landmarks for the planning problem $P = \langle D_{\text{other}}, \mathrm{u} \rangle$ or $P = \langle D_{\text{user}}, \mathrm{u} \rangle$. *Landmarks* have been successfully used in deriving heuristics in Plan Recognition (Vered et al., 2018) and generating alternative plans (Bryce, 2014). We define *attack landmarks* ($\mathcal{L}_u$) to be those predicates which must be true to reach u. We compute the percentage of active attack landmarks in the current state ($\mathcal{L}_{ac}$), where $\mathcal{L}_{ac} = \frac{l}{|\mathcal{L}_u|}$. In Figure 5.3, $l = 4$ and $\mathcal{L}_{ac} = 4/10 = 0.4$. For each presented action, the Intervention Graph is generated and features are computed, producing the corresponding feature vector. Landmarks are computed apriori.

5.5.3 Plan Space Sampling and Plan Distance Metrics

Extracting Intervention Graph features can be intractable when the graph is large. Therefore, for our second method for implementing $plan(o_i, g)$ we define an additional set of features, called *Sampled Features* by sampling the plan space for the observer. If the Intervention Problem is defined for a single user, we sample the observer's plan space by using an automated planner to find solutions for $P_{\text{observer}} = \langle F_{\text{user}}, A_{\text{user}}, s_0, g \rangle$, where $g \in \{\mathrm{d}, \mathrm{u}\}$. If the Intervention Problem is defined for a user and a competitor, we sample the observer's plan space by using an automated planner to find solutions for $P_{\text{observer}} = \langle F_{\text{user}} \cup F_{\text{other}}, A_{\text{user}} \cup A_{\text{other}}, s_0, g \rangle$, where $g \in \{\mathrm{d}, \mathrm{u}\}$. Note that in this method we omit $\mathrm{u} \cup \mathrm{d}$ for generating plans. Sampled plans are generated with the Top-K planner (Riabov et al., 2014). We estimate the Risk and Desirability using plan distance metrics. The intuition is that if the actor is executing an unsafe plan, then that plan should be more similar to a sample of unsafe plans, compared to a sample of safe plans.

For each presented action, the observer computes plan distances between a reference plan (π') and sampled plans (Π'') for both u and d. We generate the observation compatible plan by concatenating the observation history with the optimal plan that reaches u (and d) to produce π' (see (Vered & Kaminka, 2017)).

We use the Top-K planner with K=50 to sample the plan space. We use Action Set Distance (ASD), State Sequence Distance (SSD), Causal Link Distance (CLD) (Nguyen et al., 2012), Generalized Edit Distance (GED) for sequences of states and actions (Sohrabi, Riabov, Udrea, & Hassanzadeh, 2016) to measure the distances between the reference plan and the sampled plans for d and u. When an action is presented, π' is computed. Then, observation compatible Top-K plans are produced for u and d separately. The medians of ASD, CLD and SSD, minimum remaining actions to u and d, minimum action GED and state GED are computed for u and d for all ⟨reference, sample⟩ pairs. Finally, we also compute the Landmark Completion Heuristic proposed by (Pereira et al., 2017). This produces the Sampled Feature vector for the presented action. Algorithm 4 shows the pseudo-code for computing the Sampled Feature vector.

5.5.4 Learning Intervention

We train a classifier to categorize the presented observation o_i into two classes: (Yes) indicating intervention is required and (No) indicating otherwise. We chose Naive Bayes, K-nearest neigh-

Algorithm 4 Build Sampled Feature Vector

```
Require: $D$, $s$, u, d
 1: $i = 0; s_i \leftarrow s_0$
 2: $prefix, suffix, \Pi'', \mathcal{V} \leftarrow \varnothing$
 3: procedure SAMPLEDSUFFIXES($D_1, s, \mathrm{u}, \mathrm{d}, O$)
 4:     for $o \in O$ do
 5:         $prefix \leftarrow prefix + o$
 6:         $s_{i+1} \leftarrow ((s_i \setminus Del(o)) \cup Add(o))$
 7:         for $g \in \{\mathrm{u}, \mathrm{d}\}$ do
 8:             $suffix \leftarrow OptimalPlan(s, g)$
 9:             $\pi' \leftarrow prefix + suffix$
10:             $\Pi'' \leftarrow$ Observation compatible Top-K plans for $g$
11:             $v_1 \leftarrow$ MedianActionSetDist($\pi', \Pi''$)
12:             $v_2 \leftarrow$ MedianCausalLinkDist($\pi', \Pi''$)
13:             $v_3 \leftarrow$ MedianStateSequenceDist($\pi', \Pi''$)
14:             $v_4 \leftarrow$ MinimumRemainingDistToState ($g, \Pi''$)
15:             $v_5 \leftarrow$ MinimumActionGED ($\pi', \Pi''$)
16:             $v_6 \leftarrow$ MinimumStateGED ($\pi', \Pi''$)
17:             $\mathcal{V}(o) \leftarrow [v_1, v_2, v_3, v_4, v_5, v_6]$
18:         $v_7 \leftarrow$ ComputeLandmarkCompletionHeuristic (u)
19:         $\mathcal{V}(o) \leftarrow \mathcal{V}(o) + \{v_7\}$
```

bors, decision tree and logistic regression classifiers from Weka [1]. Given observations labeled as Yes/No and corresponding feature vectors as training examples, we train the classifiers with 10-fold cross validation. The trained model is used to predict intervention for previously unseen Intervention Problems. Attribute selected classifiers filter the feature vector to only select critical features. This step reduces complexity of the model, makes the outcome of the model easier to interpret, and reduces over-fitting.

We generated training data from twenty Intervention Problems using the benchmark domains. We used the Blocks World domain to model competitive Intervention Problems and Ferry, EasyIPC and Navigator domains to model Standard Intervention Problems. We restricted the number of observation traces per Intervention Problem to 100 for training the classifiers.

The decision tree classifier is tuned to pruning confidence=0.25 and minimum number of instance per leaf=2. The K-nearest neighbor classifier is tuned to use k=1 and distance mea-

[1] `http://www.cs.waikato.ac.nz/ml/weka/`

sure=Euclidean. The logistic regression classifier is tuned for ridge parameter = 1.0E-8. The Naive Bayes classifier is tuned with the supervised discretization=True.

5.6 Evaluating Intervention Recognition

As the baseline, we compare the learning based intervention accuracy to three state-of-the-art Plan Recognition algorithms from the literature. Our evaluation focuses on two questions: (1) Using domain-independent features indicative of the likelihood to reach u from current state, can the observer correctly recognize directly and indirectly contributing suffixes to prevent the user from reaching u? and (2) How does the learning approach perform against state-of the-art Plan Recognition? To address the first question, we evaluated the performance of the learned model on unseen Intervention Problems.

The benchmarks consist of Blocks-words, IPC Grid, Navigator and Ferry domains. For the **Blocks-words** domain, we chose word building problems. The user and the competitor want to build different words with some common letters. The problems in **Blocks-1** model intervention by identifying the direct contributors (a_{crit}), whereas the problems in **Blocks-2** model intervention by identifying the indirect contributors (q_{crit}) in the Blocks-words domain. In the **IPC grid** domain, the user moves through a grid to get from point A to B. Certain locked positions on the grid can be opened by picking up keys. In the **Navigator** domain, the user moves from one point on a grid to another. In IPC Grid and Navigator domains, we designated certain locations on the grid as traps. The goal of the user is to navigate to a specific point on the grid without passing through the trap. In the **Ferry** domain, a single ferry moves cars between different locations. The ferry's objective is to transport cars to specified locations without using a port, which has been *compromised*.

To evaluate our trained classifiers, we generate 3 separate test problem sets with 20 problems in each set (total of 60) for the benchmark domains. The test problems differ from the training data. The three test problems vary the number of blocks in the Blocks Words domain, size of the grid (Navigator, IPC-Grid), accessible and inaccessible paths on the grid (Navigator, IPC-Grid), and properties of the artifacts in the grid (IPC-Grid). Each test problem includes 10 observation traces (total of 600 test cases). We designed the Blocks-1, IPC-Grid, Ferry, and Navigator problems specifically considering desirable/undesirable goal pairs that are difficult to disambiguate using existing plan recognition algorithms. We designed the problems in Blocks-2 domain to include problems that will be easier to solve by existing plan recognition algorithms.

We define true-positive as the classifier correctly identifying the presented action to be in a_{crit} or q_{crit}. True-negative is an instance where the classifier correctly identifying an action as not belonging to a_{crit} or q_{crit}. False-positives are instances where classifier incorrectly identifies an action as belonging to a_{crit} or q_{crit}. False-negatives are instances where the classifier incorrectly identifies the presented action as not belonging to a_{crit} or q_{crit}. Naturally, our test observation traces contain a large number of negatives. To offset the bias introduced to the classifier by the class imbalance, we report Matthews correlation coefficient (MCC) because it gives an accurate measure of the quality of a binary classification while taking into account the different class sizes. We also report the F-score $= \frac{tp}{tp+1/2(fp+fn)}$ for the classifiers, where tp, fp, fn are the number of true positives, false positives and false negatives respectively.

We implemented three state-of-the art Plan Recognition algorithms to compare the accuracy of intervening by Plan Recognition to the proposed learning based intervention. We selected Ramirez and Geffener's probabilistic Plan Recognition algorithm (Ramírez & Geffner, 2010) (both the satisficing, and optimal implementations) and the Goal Recognition with Goal Mirroring algorithm (Vered et al., 2018). To generate satisficing plans we used the Fast Downward planner with the FF heuristic and context-enhanced additive heuristic (Helmert, 2006). To generate the optimal cost plans we used the HSP planner (Bonet & Geffner, 2001). For each presented action, the observer solves a Plan Recognition problem using each approach. We assumed uniform priors for over u and d. If u is the top ranked goal for the presented action, then it is flagged as requiring intervention. The assumption is that these algorithms must also be able to correctly identify u as the most likely goal for the actions in a_{crit} and q_{crit}. We used the same test data to evaluate accuracy.

Table 5.4: F-score and MCC for predicting intervention using Intervention Graph and the Plan Space Sampling methods for Naive Bayes and Decision Tree classifiers

Domain	Naive Bayes						Decision Tree					
	Test Set 1		Test Set 2		Test Set 3		Test Set 1		Test Set 2		Test Set 3	
	F-score	MCC	F-score	MCC	F-score	MCC	F-score	MCC	F-score	MCC	F-score	MCC
Intervention Graph Method												
Blocks-1	1	1	1	1	1	1	1	1	1	1	1	1
Blocks-2	1	1	1	1	1	1	1	1	1	1	1	1
EasyIPC	1	1	1	1	1	1	1	1	1	1	1	1
Ferry	1	1	1	1	1	1	1	1	1	1	1	1
Navigator	1	1	1	1	.99	.99	.87	.87	.72	.74	.90	.90
Plan Space Sampling Method												
Blocks-1	.25	.33	.25	.33	.25	.33	.25	.33	.25	.33	.25	.33
Blocks-2	1	1	1	1	1	1	1	1	0.99	0.99	1	1
EasyIPC	1	1	1	1	1	1	1	1	1	1	1	1
Ferry	.34	.33	.32	.31	0.02	-.004	.25	.28	.24	.23	.86	.86
Navigator	1	1	1	1	1	1	.62	.65	1	1	1	1

Tables 5.4 and 5.5 shows that the classifiers trained with features from the Intervention Graph achieve high accuracy for all the domains when predicting intervention for both identifying actions in a_{crit} and q_{crit}. The MCC value shows that the imbalance in class sizes does not bias the classifier. Low false positives and false negatives suggest that the user will not be unnecessarily interrupted. As expected performance degrades when we use a sampled plan space to derive features. However, the features derived from sampling the plan space produce equally good classifiers compared to the intervention graph method, when modeling intervention by identifying actions in a_{crit} and q_{crit} for the Blocks-word problems. For the Intervention Problems modeled using the benchmark domains, we were able to find that at least one of the selected classifiers responded with very high F-score when trained using plan similarity features. The exception to this pattern was the Intervention Problems modeled using the Ferry domain.

Features derived from the Intervention Graph accurately recognize actions in a_{crit} and q_{crit} where the user has limited options available to reach the desirable goal while avoiding the undesirable state. Thus the classifiers perform well in recognizing critical actions in new problems. The sampled features rely on the learning algorithm to produce accurate results when predicting intervention.

Table 5.5: F-score and MCC for predicting intervention using Intervention Graph and Plan Space Sampling methods for logistic regression and k-nearest neighbor classifiers

Domain	Logistic Regression						K-Nearest					
	Test Set 1		Test Set 2		Test Set 3		Test Set 1		Test Set 2		Test Set 3	
	F-score	MCC	F-score	MCC	F-score	MCC	F-score	MCC	F-score	MCC	F-score	MCC
Intervention Graph Method												
Blocks-1	1	1	1	1	1	1	1	1	1	1	1	1
Blocks-2	1	1	1	1	1	1	1	1	1	1	1	1
EasyIPC	.88	.87	.88	.87	.86	.86	1	1	1	1	1	1
Ferry	1	1	1	1	1	1	1	1	1	1	1	1
Navigator	1	1	1	1	.99	.99	1	1	.96	.96	.99	.99
Plan Space Sampling Method												
Blocks-1	.25	.33	.25	.33	.25	.33	1	1	1	1	1	1
Blocks-2	1	1	1	1	1	1	1	1	1	1	1	1
EasyIPC	.64	.63	.46	.44	.67	.66	.05	-.04	.04	-.03	.05	-.02
Ferry	.31	.32	.23	.22	1	1	.33	.40	.13	.15	.81	.82
Navigator	.60	.59	.98	.94	.97	.97	.61	.65	1	1	1	1

Table 5.6: F-score and Matthews Correlation Coefficient (MCC) for recognizing intervention using probabilistic goal recognition (RG) algorithm proposed by Ramirez and Geffener. RG (LAMA) is when a satisficing planner (LAMA) is used to generate the plans for probabilistic goal recognition. RG (HSP) is when an optimal (HSP) planner is used to generate the plans for probabilistic goal recognition. Goal mirroring (GM) implements the recognition algorithm proposed by Vered et al. and uses the JavaFF planner to generate the plans. For the Intervention problems in the Navigator domain, the true-negative, false negative rates were 100% each. Therefore, F-score and MCC are not reported.

Domain	Test Set 1						Test Set 2						Test Set 3					
	RG (LAMA)		RG (HSP)		GM		RG (LAMA)		RG (HSP)		GM		RG (LAMA)		RG (HSP)		GM	
	F-score	MCC	F-score	MCC	F-score	MCC	F-score	MCC	F-score	MCC	F-score	MCC	F-score	MCC	F-score	MCC	F-score	MCC
Blocks-1	.38	.45	.38	.45	.36	.43	.43	.49	.43	.49	.39	.45	.40	.47	.40	.47	.38	.45
Blocks-2	1	1	.90	.90	1	1	1	1	.90	.90	1	1	1	1	.90	.90	1	1
EasyIPC	.13	.05	.13	.05	.10	.01	.21	.17	.18	.13	.12	.06	.23	.19	.22	.19	.14	.09
Ferry	.17	.18	.22	.20	.10	.08	.22	.23	.11	.06	.15	.09	.15	.17	.47	.52	.21	.34

Comparing the results in Table 5.6, learning methods outperform existing Plan Recognition algorithms when predicting intervention. The algorithms we selected clearly struggled to predict intervention in the Navigator domain producing many false positives and false negatives. These results suggest that although we can adopt existing Plan Recognition algorithms to identify when the user needs intervention, it also produces a lot of false negatives and false positives in correctly identifying which state (desirable/undesirable) is most likely given the observations, especially when the desirable and undesirable plans have a lot of overlap. Comparing recognition accuracy of Blocks-1 and Blocks-2 problems using plan recognition algorithms to intervene, we see that when the undesirable state develops over a long period of time, the plan recognition algorithms recognize the undesirable plan with high accuracy. In other words, if the d and u are sufficiently distinct and are far apart in the state space, intervention by using plan recognition algorithms are as effective as our proposed learning based intervention methods in most cases. When the desirable and undesirable states are closer, existing plan recognition algorithms produce many false positives/negatives.

This is unhelpful specially considering human users because, when the intervening agent produces many false alarms and/or miss critical events that must be intervened, the human users may get frustrated and turn off intervention. Therefore, the learning approach is better suited for intervention because the observer can target specific critical actions and at the same time the user is given some freedom to pursue a desirable goal.

5.6.1 Processing Times

Because the Intervention problem is defined for online environments, we now report the processing time comparison among the two proposed learning based Intervention algorithms and intervention using existing plan recognition algorithms. The experiments were run in an Intel Core i7 CPU at 1.30GHz x 8 machine running on Ubuntu 20.04LTS. We compute two evaluation metrics for the processing time comparison. **The total processing time** (Q) is the CPU time in millisec-

Table 5.7: Total processing time in **seconds** (Q) for all Intervention problems in set 1, 2 and 3 and the mean processing time in **milliseconds** for each incrementally revealed observation ($\overline{Q}$) for predicting intervention with Naive Bayes and Decision Tree classifiers using features from the Intervention Graph and the Plan Space Sampling methods

Domain	Naive Bayes						Decision Tree					
	Test Set 1		Test Set 2		Test Set 3		Test Set 1		Test Set 2		Test Set 3	
	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)
Intervention Graph Method												
Blocks-1	42.4	64	39.9	60	42.5	64	40.6	61	39.3	59	42.0	63
Blocks-2	40	60	38.1	57	38.4	58	40.4	61	38.0	57	40.2	60
EasyIPC	7519.9	3484	14858.0	5371	805.3	403	7615.4	3528	15842.8	5727	802.6	402
Ferry	7992.5	3748	72.1	119	10656.9	4492	7921.4	3715	62.0	103	107945.0	4550
Navigator	1530.0	1969	880.4	896	77.4	102	1694.4	2180	882.2	898	84.6	111
Plan Space Sampling Method												
Blocks-1	409.5	620	405.6	614	402.6	610	338.3	512	365.1	553	365.9	554
Blocks-2	357.6	541	353.5	534	350.5	531	392.7	595	401.4	606	3743.4	567
EasyIPC	2274.8	1054	4607.1	1665	2100.2	1053	2324.1	1076	5023.9	1816	2064.8	1035
Ferry	1479.8	694	195.0	323	1451.0	611	1564.9	733	219.1	364	1697.8	715
Navigator	1131.7	1456	1231.4	1254	373.4	493	1050.6	1352	1183.2	1204	369.7	489

onds taken to process all the 20 Intervention problems in a test set. **The mean processing time** ($\overline{Q}$) is the CPU time in milliseconds taken to return the intervention decision for one incremental observation reveal. It is given by the equation: $\overline{Q} = \frac{Q}{\text{number of observations in the test set}}$. Tables 5.7 and 5.8 show the Q and $\overline{Q}$ values for returning the intervention decisions using the classifiers trained with the Intervention Graph features and the plan distance features. When the classifiers are trained with the Intervention Graph features the smallest $\overline{Q}$ (< 70 milliseconds) are reported for the Blocks-words domain. The largest $\overline{Q}$ values are reported for the EasyIPC and the Ferry domain (< 5.5 seconds). When the classifiers are trained with the plan distance features, $\overline{Q}$ is larger compared to the previous case. However, the values for $\overline{Q}$ are still lower (< 620 milliseconds) compared to the other domains. The largest $\overline{Q}$ are reported for the EasyIPC and the Ferry domains (< 1.8 seconds). There is a large range of the total processing times (maximum Q - minimum Q) among the Intervention problems from different domains.

In contrast, when using plan recognition algorithms for intervention, $\overline{Q}$ is smaller compared to the $\overline{Q}$ in learning based intervention. For the Test Set 1, the smallest $\overline{Q}$ is reported for the EasyIPC problems when using probabilistic plan recognition using the optimal plans (< 42 milliseconds).

Table 5.8: Total processing time in **seconds** (Q) for all Intervention problems in set 1, 2 and 3 and the mean processing time in **milliseconds** for each incrementally revealed observation ($\overline{Q}$) for predicting intervention with logistic regression and k-nearest neighbor classifiers using features from the Intervention Graph and the Plan Space Sampling methods

Domain	Logistic Regression						K-nearest					
	Test Set 1		Test Set 2		Test Set 3		Test Set 1		Test Set 2		Test Set 3	
	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)
Intervention Graph Method												
Blocks-1	40.3	61	38.8	58	42.4	64	40.8	61	39.8	60	41.4	62
Blocks-2	40.7	61	38.9	58	39.4	59	41.3	62	39.5	59	41.6	63
EasyIPC	6955.0	3222	15309.2	5534	920.8	461	7159.8	3317	14821.0	5358	837.0	419
Ferry	7586.2	3558	54.3	90	9598.3	4046	8116.9	3807	57.7	95	10029.4	4228
Navigator	1615.4	2079	868.6	884	91.7	121	1441.7	1855	881.9	898	93.2	123
Plan Space Sampling Method												
Blocks-1	390.3	591	405.8	614	398.4	603	376.3	570	372.2	563	372.4	564
Blocks-2	345.4	523	342.1	516	338.1	512	388.8	589	372.2	562	355.2	538
EasyIPC	2286.7	1059	4596.6	1661	1898.5	952	2528.8	1171	4702.0	1699	1928.9	967
Ferry	1531.9	718	210.0	348	1562.0	658	1549.2	726	212.2	352	1574.0	663
Navigator	1095.7	1410	1209.9	1232	380.4	503	1076.1	1384	1208.6	1230	377.0	498

Table 5.9: Total processing time in **seconds** (Q) for all Intervention problems in set 1, 2 and 3 and the mean processing time in **milliseconds** for each incrementally revealed observation ($\overline{Q}$) for recognizing undesirable states with the algorithms: probabilistic goal recognition using satisficing (RG-LAMA) and optimal (RG-HSP) plans and goal mirroring (GM) using the JavaFF planner.

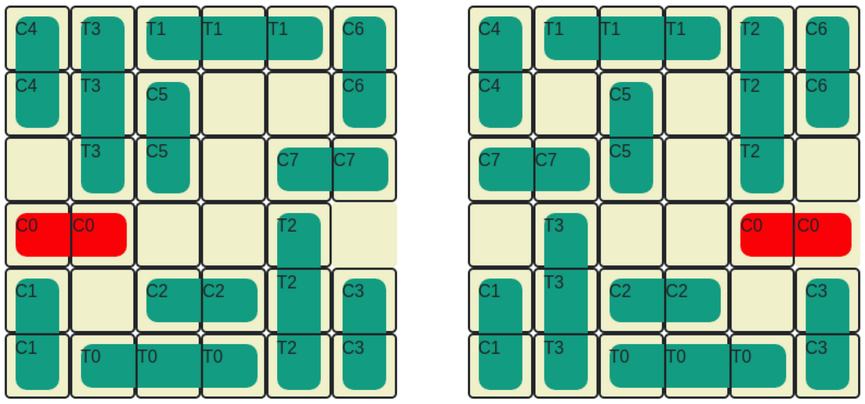
Domain	Test Set 1						Test Set 2						Test Set 3					
	RG (LAMA)		RG (HSP)		GM		RG (LAMA)		RG (HSP)		GM		RG (LAMA)		RG (HSP)		GM	
	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)	Q (s)	$\overline{Q}$ (ms)
Blocks-1	147.9	224	65.8	99	151.6	229	147.1	222	60.1	91	148.9	225	148.1	224	63.3	95	152.0	230
Blocks-2	145.6	220	108.9	164	153.6	232	144.5	218	113.2	171	155.0	234	145.6	220	107.5	162	144.9	219
EasyIPC	474.0	219	91.5	42	450.6	208	599.5	216	418.3	151	549.0	198	427.2	214	70.6	35	407.9	204
Ferry	440.1	206	3266.6	1532	366.3	171	117.2	194	75.4	125	97.5	162	480.5	202	344.1	145	407.0	171

The largest $\overline{Q}$ is reported for the Ferry domain when using probabilistic plan recognition using the optimal plans (1.5 seconds), also in Test Set 1. Most of $\overline{Q}$ values are less than 232 milliseconds. The range of the total processing times (maximum Q - minimum Q) among the Intervention problems from different domains is smaller compared to the range of the total processing times using the learning based intervention. While plan recognition algorithms *sometimes* return an intervention decision more quickly than the learning based intervention, their accuracy, precision, and recall is substantially lower than learning based intervention.

5.7 Human-Aware Intervention

The Human-aware Intervention Problem is a variant of the single-user intervention, so the history H consists of only accepted user actions. However, as mentioned in Section 5.4.2, the set of projections will be empty, that is $\mathcal{X}_{\Diamond} = \emptyset$ because planning is less accurate for projecting what the human user may do. This means the observer must emphasize analysis of H. Instead of relying on projections in $\mathcal{X}_{\Diamond}$, the observer *learns* the function $intervene_i$. That is, at the presented action o_i it considers only H, d, and u and uses a trained machine learning algorithm to decide whether to intervene.

We introduce the Rush Hour puzzle as a benchmark in order to study how human users solve the puzzle as a planning task that requires intervention. We begin with the formal definitions of the Rush Hour puzzle. Next, we translate the Rush Hour problem into a STRIPS planning task and through a human subject study, we allow human users solve the planning problem and collect observation traces. We formally define the Human-aware Intervention problem and propose a solution that uses machine learning to learn properties of H to determine whether or not intervention is required. The observer for Human-aware Intervention should offer different levels of freedom to the user. At the lowest level of freedom, the observer will intervene just before the undesirable state. At increased levels of freedom, the observer offers the user enough time to recover from the undesirable situation. Varying the level of freedom allows the observer to gradually guide the



(A) Initial game state **(B) End game state**

Figure 5.4: A Rush Hour instance

user toward d without directly giving away the complete solution. We evaluate the accuracy of our learning approach using the observation traces collected from the human subject study.

5.7.1 The Rush Hour Puzzle

The Rush Hour puzzle is a game for ages 8 and above (Flake & Baum, 2002). Figure 5.4A shows an initial puzzle configuration on a 6×6 grid, where cars (length 2) and trucks (length 3) are arranged. Vertically aligned vehicles can only move up and down and horizontal vehicles can move left and right. Vehicles can be moved one at a time and into adjacent empty spaces. The solution to the Rush Hour puzzle is a sequence of legal moves that transforms the initial board state shown in 5.4A to the goal state in 5.4B. For the puzzle shown in Figure 5.4, the shortest solution has 21 moves, if the number of moves is considered as the optimizing criteria. If optimized for the number of vehicles moved, the puzzle can be solved optimally by moving only 8 vehicles. It is important to note that one can obtain different "optimal solutions" depending on whether the number of moves is minimized, or if the number of vehicles moved is minimized. Humans tend not to clearly make this distinction when playing the game.

We adopt the formal definition of a Rush Hour instance from (Flake & Baum, 2002).

Definition 11. *A* Rush Hour instance *is a tuple* $\langle w, h, x, y, n, \mathcal{V}\rangle$ *such that:*

- $(w, h) \in \mathbb{N}^2$ *are the grid dimensions. In the standard version,* $w = h = 6$

- $(x, y), x \in \{1, w\}$ *and* $y \in \{1, h\}$ *are the coordinates of the exit, which must be on the grid perimeter.*

- $n \in \mathbb{N}$ *the number of non-target vehicles*

- $\mathcal{V} = \{v_0, \ldots, v_n\}$ *is the set of* $n+1$ *vehicles comprised of cars* $(\mathcal{C})$ *and trucks* $(\mathcal{T})$*. Note that* $|\mathcal{V}| = |\mathcal{C}| + |\mathcal{T}|$

  $v_i \in \mathcal{C}$ *is identified as* $\{C_0, \ldots, C_l\}$*, where* $l = |\mathcal{C}| - 1$ *and* $v_i \in \mathcal{T}$ *is identified as* $\{T_0, \ldots, T_m\}$*, where* $m = |\mathcal{T}| - 1$.

  A vehicle is a tuple $v_i = \langle x_i, y_i, o_i, s_i \rangle$*, where* $(x_i, y_i) \in \mathbb{N}^2$ *are the vehicle coordinates,* $o_i \in \{N, E, S, W\}$ *is the vehicle orientation for North, East, South, West,* $s_i \in \{2, 3\}$ *is the vehicle size and* C_0 *is the target vehicle.*

(Flake & Baum, 2002) also defines the *solution* to a Rush Hour instance as a sequence of m moves, where each move consists of a vehicle identifier i, a direction that is consistent with the initial orientation of v_i, and a distance. Each move, in sequence must be consistent with itself and with the configuration prior to the move. Further, in order to move a distance d the configuration must be consistent for all d' such that, $0 \leqslant d' \leqslant d$ (i.e., a vehicle cannot jump over other vehicles on its path).

5.7.2 The Rush Hour Puzzle as a STRIPS Planning Task for Intervention

We study how human users approach solving a cognitively engaging problem as a planning task and evaluate whether we can use domain-specific features to predict intervention. To collect realistic data, it is critical that the human users were willing to participate in the task. The Rush Hour puzzle addresses this requirement well, as evidenced by the feedback about the task we received from the demographic survey. See Section E.3 in the Appendix for details. Although the

Rush Hour puzzle is a game, the environment is rich enough to simulate undesirable consequences and also offer the users a task that is challenging enough.

We translate the Rush Hour puzzle in Definition 11 into a grounded STRIPS planning task $P = \langle F, A, s_0, G\rangle$ as follows:

- $F = \{$ $\{\forall C_i \in \mathcal{C}$, (car $?C_i$)$\}$, $\{\forall T_i \in \mathcal{T}$, (truck $?T_i$)$\}$ - for the vehicles,
  $\{\forall v_i \in \mathcal{C} \cup \mathcal{T}$ and $l_i \in \{(x, y)|x \in [1..w], y \in [1..h]\}$, (at $?v_i$ $?l_i$)$\}$ - for the vehicle positions,
  $\{\forall v_i \in \mathcal{C} \cup \mathcal{T}$ and $d_i \in \{NS, SN, EW, WE\}$, (face $?v_i$ $?d_i$)$\}$ - for direction of vehicles (North to South (down), South to North (up), East to West (left), West to East (right) respectively),
  $\{\forall l_i \in \{(x, y)|x \in [1..w], y \in [1..h]\}$, (free $?l_i$)$\}$ - for open positions,
  $\{\forall l_i, l_j \in \{(x, y)|x \in [1..w], y \in [1..h]\}$ and $d_i \in \{NS, SN, EW, WE\}$, (next $?d_i$ $?l_i$ $?l_j$)$\}$ - for direction of the adjacent locations) $\}$

- $A = \{$`move-car` = $\langle$pre(`move-car`), add(`move-car`), del(`move-car`)$\rangle \subseteq F$,
  `move-truck` = $\langle$pre(`move-truck`), add(`move-truck`), del(`move-truck`)$\rangle \subseteq F\}$

- $s_0 \subseteq F$

- $G = \{$(at C_0 l_i)(at C_0 l_j)$\}$, where $l_i = (w, 3)$ and $l_j = (w - 1, 3)$

In order to configure the Rush Hour STRIPS planning task for intervention, we introduce an undesirable state, u by designating one vehicle as *forbidden*. The post-conditions of any action that moves the forbidden vehicle satisfies u. The puzzle can be solved without moving the forbidden vehicle. Therefore, moving the forbidden vehicle is also an unnecessary action, indicating that the user is exploring an unhelpful region in the state space. Here, intervention is required to guide the user toward exploring more helpful regions in the state space. In the Figure 5.4A, the forbidden vehicle is C_2. If the user moves C_2 to the left, then the board state satisfies u. The user's goal $\mathrm{d} = G$. The vehicle movement constraint introduced by the presence of the forbidden vehicle adds an extra level of difficulty to the user's planning task.

The Rush Hour problem is also unique in that the observer is more focused on states than actions. Recall that a history $H = (o_1[s_1], o_2[s_2], \ldots, o_{i-1}[s_H])$ is a sequence of previously observed actions, which started from s_0 with the implied resulting states in brackets. For Rush Hour, the observer relies on those implied states instead of just the actions. For simplicity in notation, we present intervention in terms of states, although it is easy to map between actions and states because of the deterministic state transition system of the planning model.

5.7.3 Domain-Specific Feature Set

For the observer to learn $intervene_i$, we develop a set of domain-specific features for the Rush Hour problem. We want the feature set to capture whether the user is advancing toward d by making helpful moves, or whether the user currently exploring a risky part in the state space and getting closer to u. We hypothesize that the behavior patterns extracted from H as features have a correlation to the event of the user moving the forbidden vehicle.

Features Based on State

The features based on state analyze the properties of the sequence of state transitions in H from s_0 to s_H $([s_0], [s_1], [s_2], \ldots, [s_H])$. Specifically, we look at the *mobility* of the objects: target vehicle (C_0), the forbidden vehicle and the vehicles adjacent to the target and the forbidden vehicles. We use the state features associated with the target vehicle to measure how close the user is to d. The state features associated with the forbidden car evaluate how close the user is to triggering u.

We manually examined the solutions produced in the human subject experiment (described later) to identify common movement patterns. Our analysis revealed that if the user was moving vehicles adjacent to the forbidden vehicle in such a way that the forbidden vehicle was freed, most users ended up moving the forbidden vehicle. Therefore, by monitoring the state changes occurring around the forbidden vehicle, we can estimate whether the user will end up moving the forbidden vehicle or not (i.e., trigger u). Similarly, state changes occurring on the target car's path to the exit,

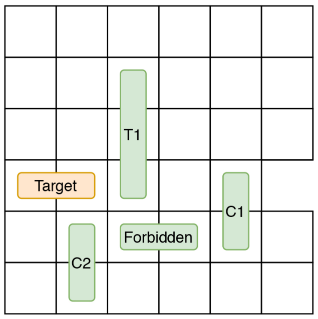


Figure 5.5: Blocker vehicles

for example, the moves that result in reducing the number of vehicles blocking the target car is considered to be helpful to move the state closer to d.

We refer to the vehicles adjacent to the target and the forbidden vehicles as *blockers* and introduce two additional object types to monitor mobility: *target car blockers* and *forbidden car blockers*. Figure 5.5 illustrates an example state. The target car's path is blocked by two vehicles C_1 and T_1. Therefore, target car blockers $= \{C_1, T_1\}$. We only consider the vehicles that are between the target car and the exit cell as target car blockers because, only those vehicles are preventing the target car from reaching d. The forbidden vehicle's movement is blocked by two vehicles C_1 and C_2. Therefore, forbidden car blockers $= \{C_1, C_2\}$. We now describe the features based on state that are used to predict intervention in Human-aware Intervention Problems.

- `blocks`: number of times a move increased the number of cars blocking the target car's path
- `frees`: number of times a move freed up empty spaces around the forbidden vehicle
- `freebci`: number of times the number of empty spaces around the forbidden vehicle blockers increased
- `freebcd`: number of times the number of empty spaces around the forbidden vehicle blockers decreased
- `freegci`: number of times the number of empty spaces around the target car blockers increased

- `freegcd`: number of times the number of empty spaces around around the target car blockers decreased
- `mgc`: mean number of empty spaces around the target car blockers
- `mbc`: mean number of empty spaces around the forbidden car blockers
- `reset`: number of times the current move changed the state back to the initial puzzle configuration

Features Based on User Actions

The features based on user state analyze the properties of the sequence of actions from o_1 to o_{i-1} in H $(o_1, o_2, \ldots, o_{i-1})$ We follow the same manual analysis of solutions produced in the human subject experiment to identify common movement patterns. We found that the users who produce unsafe solutions often made unhelpful moves such as moving the same vehicle back and forth many times in quick succession, causing their solution to be longer compared to a safe solution. We statistically verified whether the relationship between the solution length and the number of forbidden vehicle moves is significant for the human subject data using Spearman's Rank Correlation Coefficient. The test showed that the relationship is significant (p-value < 0.05). See Section E.4 in the Appendix for a summary of raw data.

Similarly, we observed that comparing the number of moves of the user's solution to an optimal solution produced by an automated planner is helpful in identifying whether the user is moving away from d or making progress. In order to verify this observation, we use the HSP planner (Bonet & Geffner, 2001) to find cost optimal solutions for the Rush Hour planning tasks (see Section 5.7.2) used in the human subject experiment. We statistically verified that the relationship between the length difference between the user's solution and the optimal solution found by an automated planner, and the number of forbidden vehicle moves is significant using Spearman's Rank Correlation Coefficient (p-value < 0.05).

Thus, we conclude that features derived from the length and number of backtracking moves in H can be used to predict when the user is getting close to $\mathtt{u}$. We introduce an unhelpful move called the *h-step backtrack*, which is a move that takes the state back to a previously seen state by h number of steps (i.e., an undo operation). When deriving the feature to capture backtracking moves, we only consider $h = 1$, which asks the question did the observation o_{i-1} undo the effect of the observation o_{i-2}?

We now describe the features based on actions that are used to predict intervention in Human-aware Intervention Problems.

- `len`: number of moves in H
- `len-opt`: difference of the number of moves in H and the number of moves in the safe optimal solution produced by an automated planner for the same planning task.
- `backtracks`: number of 1-step backtrack actions in H
- `first`: number of moves until the forbidden vehicle was moved for the first time in H
- `prop`: number of moves until the forbidden vehicle was moved for the first time in H divided by the number of moves in the safe, optimal solution produced by an automated planner for the same planning task.
- `moved`: number of vehicles moved in H

5.8 Evaluating Human-Aware Intervention

To evaluate the efficacy of the features based on state and features based on actions in predicting intervention for Human-aware Intervention Problems, we use actual observation traces collected from a human subject study, where human users solve Rush Hour planning tasks on a Web simulator. We generate learned models that predict intervention while offering different levels of freedom to the user. We consider three levels of freedom ($k = \{1, 2, 3\}$) for the evaluation. A model for the lowest level of freedom ($k = 1$), predicts intervention one move before the undesirable state. This configuration offers no time for the user to recover from the undesirable state. A model for the next level of freedom ($k = 2$), intervenes the user two moves before the undesirable state is satisfied and offers the user some time to take corrective action. A model for the highest level of freedom ($k = 3$), intervenes the user three moves before the undesirable state. We begin with the experiment protocol and briefly describe the findings. Next, we discuss the learning methods used to predict intervention. Finally, we discuss the accuracy of prediction compared to the Plan Recognition as Planning algorithm proposed by (Ramírez & Geffner, 2010).

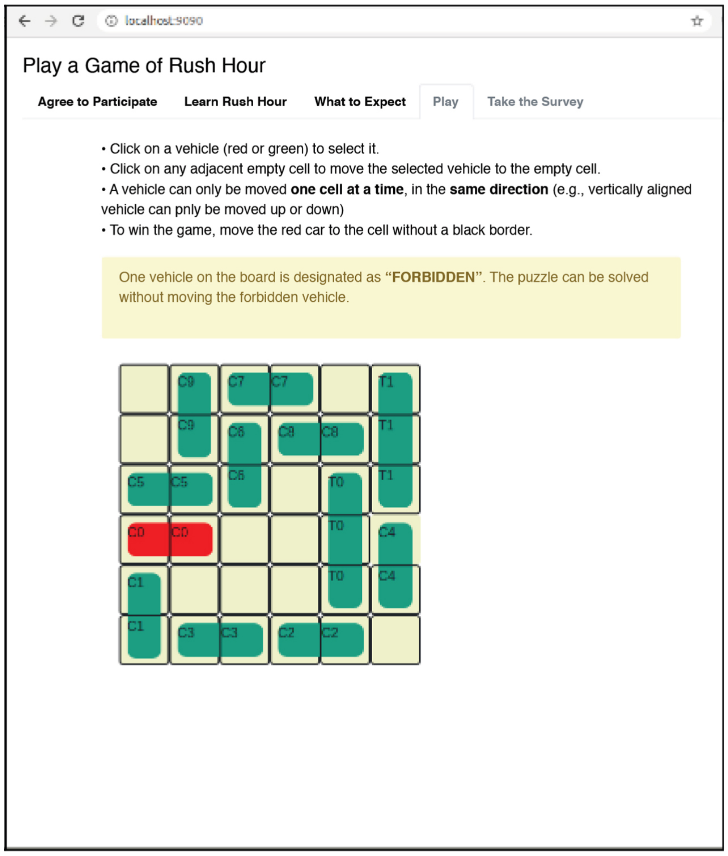
5.8.1 Rush Hour Experiment Protocol

We recruited subjects from a university student population. The sample comprised of college students in Computer Science, Psychology, Agriculture and Business majors. 136 participants completed the study. The participants were not compensated for their time. After obtaining informed consent, the participants were directed to the Web URL (`https://aiplanning.cs.colostate.edu:9080/`), which hosted the Rush Hour simulator software. Each participant was assigned to solve one randomly selected Rush Hour puzzle. We did not place any time restriction for the puzzle solving task. Participants also had the option to use an online tutorial (available on the Web simulator application) on how to play the Rush Hour puzzle. Each puzzle contained one forbidden vehicle. Once the puzzle solving task was completed, the participants were asked to complete a short demographic survey on their general puzzle solving habits. 117 of the 136 participants also completed the demographics survey.

When choosing Rush Hour puzzle instances for the human subject study, we want to carefully balance the puzzle's difficulty for a human user. Especially, considering the PSPACE-completeness of the (generalized) puzzle, we need the puzzles to be solvable by human users in a reasonable time. We used a pilot study to determine the puzzle difficulty. See Section E.1 in the Appendix. We ensured that the experiment protocol fully adhered to the Rush Hour planning task definitions (Section 5.7.2). The goal of the Rush Hour planning task (d) is clearly communicated to the user. To instill the importance of avoiding the forbidden vehicle in the user's mind, we provided an information message (yellow information bar in Figure 5.6) to inform about the presence of a forbidden vehicle without specifying the vehicle identifier. The users were also informed that the puzzle can be solved without moving the forbidden vehicle. If the user moved the forbidden vehicle, no visual cues (error messages, blocks) were given. Therefore, the specific undesirable state (u) remained hidden to the user, but they were made aware of its presence. We informed the human users that there is a forbidden vehicle and they must try to solve the puzzle without moving it to prime them toward thinking more deeply about the puzzle and *raise the aware of the undesirable state*. This technique allowed us to increase the cognitive load on the human user because it conditioned the human users to study the puzzle to guess/recognize what the forbidden vehicle. Further it primed the users to make thoughtful moves instead of making random moves and accidentally finding the solution.

This was done to prime the user into studying the puzzle thinking carefully about the moves and increase the cognitive load while solving the planning task. To simulate discrete actions, the vehicles on the board could only be moved one cell at a time. The user can click on the object to select it and move it by clicking on an empty adjacent cell. Invalid moves (vehicle dragging and jumps) are blocked and the user is notified via an alert message. We record the user's solution to the STRIPS planning task as a sequence of actions in a text file.

We use ten Rush Hour planning tasks for the experiment. For analysis purposes (see Appendix), we separate the ten puzzles into three groups by the position of the forbidden vehicle. As shown in Figure 5.6(B), type **C** has the forbidden vehicle in the **c**orner of the board. Type **E** has the forbidden vehicle on an **e**dge. Type **M** has the forbidden vehicle in the **m**iddle. The experiment uses four puzzles of type C, five puzzles of type E and one puzzle of type M.

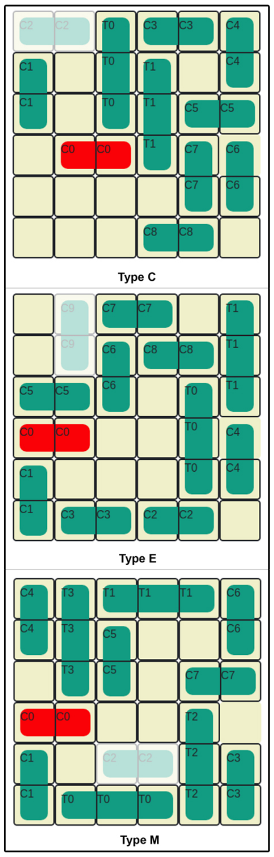

Play a Game of Rush Hour

Agree to Participate | Learn Rush Hour | What to Expect | Play | Take the Survey

• Click on a vehicle (red or green) to select it.
• Click on any adjacent empty cell to move the selected vehicle to the empty cell.
• A vehicle can only be moved **one cell at a time**, in the **same direction** (e.g., vertically aligned vehicle can pnly be moved up or down)
• To win the game, move the red car to the cell without a black border.

One vehicle on the board is designated as **"FORBIDDEN"**. The puzzle can be solved without moving the forbidden vehicle.

(A)



(B)

Figure 5.6: (A) Rush Hour planning task Web interface. The forbidden vehicle for this configuration is C_9. (B) Rush Hour puzzle configuration types with forbidden vehicles highlighted.

5.8.2 Length Distributions of Human Users' Solutions

We now compare the cost of the solutions produced by human users by dividing the solutions into two types: safe and unsafe. In this analysis, we assumed each move is unit cost, therefore the cost of the solution is equal to the number of moves. We refer to solutions that did not move the forbidden vehicle as safe and solutions that moved the forbidden vehicle as unsafe. 66 users from the total 136 solutions that involved moving the forbidden vehicle (49%). From those who moved the forbidden vehicle, 54 users moved the vehicle more than once (82%). Table 5.10 describes the summary statistics for the safe and unsafe solutions.

We see that the planning task P1 did not produce any unsafe solutions. The reason for this observation is that when examining the structure of P1 it can be seen that moving the forbidden vehicle makes the planning task unsolvable. Second, the planning task P8 did not produce any unsafe solutions. Furthermore, P8 and P6 planning tasks in type C did not produce any unsafe solutions. Human users found it difficult to avoid the forbidden car for type E planning tasks, where the forbidden car was positioned along the edge of the board. For type E planning tasks (P3, P5, P9, P10), there are more unsafe solutions than safe solutions and also the mean solution length for unsafe solutions is larger compared to the safe solutions mean. We only had 1 planning task for type M (P7), where the forbidden vehicle was placed in the middle of the board. The users who attempted P7 found it difficult to solve the planning task without moving the forbidden vehicle.

Figure 5.7 illustrates how the number of moves in users' solutions compare to a set of threshold values derived from the optimal solution for each Rush Hour planning task. We define the threshold set θ as: given the optimal number of moves α for a puzzle, $\theta = \{\alpha, 1.2\alpha, 1.4\alpha, 1.6\alpha, 1.8\alpha\}$. The letter in parenthesis indicates the puzzle group (see Figure 5.6(B) for the three puzzle types) of each puzzle.

It can be seen that human solvers' solutions to P8 were very close to the optimal solution in the number of moves. Human solvers' found it very difficult to find a solution closer to the

Table 5.10: Frequency, minimum, maximum, mean and standard deviation (SD) of the number of moves in human user solutions for the Rush Hour planning tasks P1 through P10. The letter within parenthesis next to each puzzle id indicate the puzzle type.

PID	Safe					Unsafe				
	freq	min	max	mean	SD	freq	min	max	mean	SD
P1 (E)	18	24	106	43.9	20.5	-	-	-	-	-
P2 (C)	3	44	158	99.3	57.1	8	78	378	190.3	120
P3 (E)	-	-	-	-	-	12	25	50	35.5	8.3
P4 (C)	9	23	46	30	7.1	7	25	124	67.4	33.9
P5 (E)	4	23	32	26.5	3.9	7	14	82	32.0	23.3
P6 (C)	14	22	55	29	10.3	-	-	-	-	-
P7 (M)	2	29	37	33	5.7	9	43	132	80.9	38.2
P8 (C)	18	9	12	9.3	0.8	-	-	-	-	-
P9 (E)	2	21	27	24	4.2	14	29	169.0	66.3	39
P10 (E)	-	-	-	-	-	9	44	158	81.2	39.2

optimal number of moves for P2. Users who attempted P3 and P5 found solutions shorter than the safe, optimal. Shorter solutions for these two puzzles all required the user to move the forbidden vehicles. This observation allows us to draw a conclusion that the recovery process of the Human-aware Intervention needs to aim at reducing the remaining number of moves the user has to execute to help them avoid the forbidden vehicle.

5.8.3 The Learning Methods

Our solution to the Human-aware Intervention Problem uses machine learning to predict whether u will be reached in k moves, given H, where $k = \{1, 2, 3\}$. To produce the learned models, we first partition the 136 human user solution from the experiment into training (70%) and test (30%) sets. To produce the H for a user, the user's solution is pre-processed to only include the moves until one step, two steps and three steps before the forbidden vehicle was moved for the first time. For example, in a solution $O = \{o_1, \ldots, o_i\}$, if a user moved the forbidden vehicle in step i, we generate three observation traces $O_1 = \{o_1, \ldots, o_{i-1}\}$, $O_2 = \{o_1, \ldots, o_{i-2}\}$ and $O_3 = \{o_1, \ldots, o_{i-3}\}$ corresponding to that user. Observation traces of type O_1 were used to train the model for $k = 1$, observation traces of type O_2 were used to train the model for $k = 2$

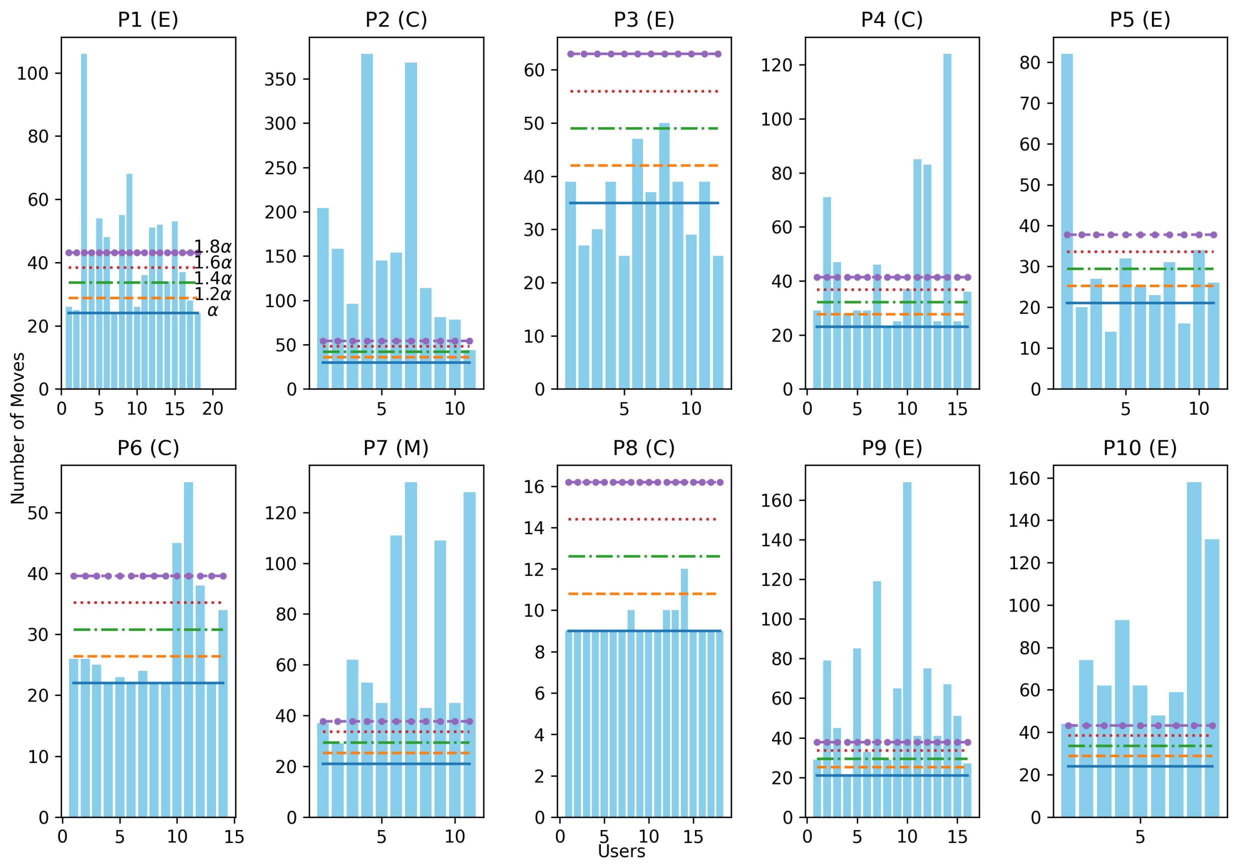


Figure 5.7: Number of moves in the users' solution compared to the optimal number of moves α, 1.2α, 1.4α, 1.6α, 1.8α for puzzles P1 through P10

and so on. Given the sequence of actions in the user's solution, the corresponding state after each move required for H is derived using the STRIPS planning model for the corresponding Rush Hour puzzle. We use the features based on state and features based on actions together to train five classifiers with 10-fold cross validation for each value of k. We explore a number of classifiers: the decision tree, K-nearest neighbor, Logistic Regression and Naive Bayes. The classifiers are used in the supervised learning mode. We summarize the parameters used in each learning method below:

- *Decision Tree:* We use the J48 classifier available on the WEKA platform (Hall et al., 2009). This classifier implements the C4.5 algorithm (Quinlan, 1993). The decision tree classifier is tuned to user pruning confidence=0.25 and the minimum number of instances per leaf=2.
- *k Nearest Neighbor (KNN):* We use this classifier with a Euclidean distance metric, considering the value $k = 1$
- *Logistic Regression:* This classifier is tuned for ridge parameter $= 1.0E - 8$.
- *Naive Bayes:* This classifier is tuned with the supervised discretization=`True`.

Human-aware Intervention Accuracy

We used the learned models on the test data set to predict whether the observer should intervene given H. In order to evaluate the classifier accuracy, we first define true-positives, true-negatives, false-positives, false-negatives for the human-aware Intervention Problem. A true-positive is when the classifier correctly predicts that u will be reached in k moves given H. A true-negative is when the classifier correctly predicts that the u will not be reached in k moves. A false-positive is when where the classifier incorrectly identifies that the u will be reached in k moves. A false-negative is when the classifier identifies that the u will not be reached in k moves, but in fact it does.

Table 5.11 summarizes the precision, recall and F-score for predicting intervention for $k = \{1, 2, 3\}$. It can be seen that the Logistic Regression classifier performs the best with high precision/recall compared to other classifiers when predicting intervention for $\{k = 1, 2\}$ When $k = 3$,

Table 5.11: Precision, Recall and F-scores for the prediction accuracy of the Human-aware Intervention learned models. Classifiers that reported the highest precision, recall and F-score are in bold. When $k = 3$, both the KNN and logistic regression classifiers have the same F-score. However, KNN has the best precision. The best recall value is reported for the logistic regression classifier. Therefore both are highlighted

Classifier	$k = 1$			$k = 2$			$k = 3$		
	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
Decision Tree	0.70	0.90	0.89	0.80	0.95	0.87	0.89	0.81	0.85
KNN	0.89	0.76	0.82	0.86	0.86	0.86	**0.95**	**0.90**	**0.93**
Logistic Regression	**0.91**	**0.95**	**0.93**	**0.87**	**1**	**0.93**	**0.91**	**0.95**	**0.93**
Naive Bayes	0.73	0.90	0.81	0.74	0.86	0.83	0.68	0.90	0.78

the Logistic Regression classifier predicts intervention with high recall but slightly lower precision. The precision of the decision tree classifier improves for higher values of k. However, the recall and F-score drop when $k = 3$. The Naive Bayes classifier reported the lowest precision compared to all other classifiers for any value of k.

Human-aware Intervention as a Plan Recognition Problem

Recall that in Section 5.2, we showed that it is possible to frame the Intervention Problem as a Plan Recognition task, where the observer models the u and d as the set of likely goals of the user. Then, we can implement the observer in the Rush Hour puzzle solving task as a probabilistic Plan Recognition agent that solves the Plan Recognition Problem $T = \langle D_{\text{user}}, \mathcal{G}, H, Prob\rangle$, where D_{user} is the planning domain, the set of likely goals $\mathcal{G} = \{\mathrm{u}, \mathrm{d}\}$, H is the observation sequence and $Prob$ is a probability distribution over $\mathcal{G}$. By compiling the observations away into the domain theory, Ramirez and Geffner showed that, it is possible to use an automated planner to find plans that are compatible with the observations and use these observation compatible plans to determine what the most likely goal and plan (Ramírez & Geffner, 2010) is for the user. They evaluated the approach on benchmark planning domains.

If the observer can recognize u as the most likely goal given the observations, then the observer must intervene at that point. We adapt the Plan Recognition as Planning (PRP) algorithm proposed by (Ramírez & Geffner, 2010), to evaluate how the observer executing PRP recognizes the most likely goal given the observation traces $O_1 = \{o_1, \ldots, o_{i-1}\}$, $O_2 = \{o_1, \ldots, o_{i-2}\}$ and

$O_3 = \{o_1, \ldots, o_{i-3}\}$, corresponding to $k = \{1, 2, 3\}$. We use the same test set from the classifier evaluation to evaluate the PRP algorithm in predicting intervention. This experiment also allows us to evaluate the PRP algorithm on plans generated by human users.

In order to evaluate the PRP accuracy, we first define true-positives, true-negatives, false-positives, false-negatives for the observer implementing the PRP algorithm. A true-positive is when the observer correctly selects u as the most likely goal given the observation traces O_1, O_2 and O_3 for $\{k = 1, 2, 3\}$ respectively. A true-negative is when the observer correctly does not select u as the most likely goal given the observation traces. A false-positive is when the observer incorrectly identifies u as the most likely goal given the observations. A false-negative is when the observer incorrectly does not select u as the most likely goal but in fact it is.

Table 5.12 summarizes the precision, recall and F-score for deciding to intervene correctly for $k = \{1, 2, 3\}$ using PRP. PRP requires goal priors as an input to the algorithm. We used the LAMA planner (Richter & Westphal, 2010) to generate satisficing plans that are compatible with the observations for PRP. Our assumption of the users' plans being satisficing is justified by the findings reported in Section 5.8.2. It shows that the majority of the users did not find optimal length solutions. In order to analyze the algorithm performance on different goal priors, we set three levels: (1) uniform goal priors, where u and d are equally likely (i.e., $P(\mathrm{u}) = P(\mathrm{d})$), (2) u is more likely and (3) d is more likely. We decided on these prior probability distributions based on the forbidden vehicle's position on initial game configurations used in the experiment. We set a higher value for u to indicate the situation where the forbidden vehicle is not blocked by other vehicles. As a result, the user will likely move it. We set d as high if the forbidden vehicle is blocked in the initial configuration, to indicate that the user will be less likely to move it. The uniform probability is the default.

It can be seen that PRP accuracy in predicting when the u will be reached is lower compared to the learned models for all levels of k. The accuracy does not change with values of k. This observation is consistent with the accuracy of the logistic regression model (i.e., the best performing learned model). When the goal priors are biased in favor of the undesirable state, the prediction

Table 5.12: Precision, Recall and F-scores for solving the Human-aware Intervention problem as a Plan Recognition problem with the Probabilistic Plan Recognition as Planning algorithm proposed by Ramirez and Geffener

Goal Priors	$k = 1$			$k = 2$			$k = 3$		
	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
Uniform	0.67	0.56	0.61	0.67	0.56	0.61	0.56	0.67	0.61
P(u) = 2×P(d)	0.69	0.61	0.65	0.69	0.61	0.65	0.69	0.61	0.65
P(d) = 2×P(u)	0.67	0.56	0.61	0.67	0.56	0.61	0.67	0.56	0.67

accuracy slightly improves. However, giving a higher prior to u contradicts with our intervention models' assumptions that the user wants to avoid the undesirable state, which implies that $P(\mathrm{u})$ must be low.

5.9 Discussion

In this chapter, we formalized a family of Intervention Problems and showed that how these problems can be solved using a combination of Plan Recognition methods and classification techniques to decide when to intervene. The Unsafe Suffix Intervention Problem uses automated planners to **project the remaining suffixes** and extract features that can differentiate unsafe remaining suffixes from the safe remaining suffixes. In contrast, the Human-aware Intervention Problem uses **only the observed history** H to extract features that can separate the solutions leading to undesirable state from the solutions that will avoid it. We compared the Unsafe Suffix Intervention and Human-aware Intervention using the state-of-art Plan Recognition approaches in the literature as the baseline and found that our learning based intervention solutions dominate the existing Plan Recognition algorithms for both benchmark Intervention Problems and a new intervention benchmark, Rush Hour.

In Unsafe Suffix Intervention, when intervention models are trained using features extracted from the Intervention Graph, all the learned models chose distance to u and Risk as the dominant features. We were not able to identify clear dominant features in learned models built with Plan Space Sampling. In Human-aware Intervention, the best performing learned model for $k = \{1, 2, 3\}$, the logistic regression classifier, selected `backtracks`, `blocks`, `frees`, `reset`, `freebci`, `freebcd`, `freegci`, `freegcd`, `mgc`, `mbc` and `moved` as the features for determining intervention.

We show that both the Unsafe Suffix Intervention Problem and the Human-aware Intervention Problem can be re-framed as Plan Recognition problems and use the recognition process to decide intervention. The results prove the feasibility of this approach. Plan recognition approaches are faster at returning a decision in some cases. However, compared to the proposed machine learning based solutions, Plan Recognition based intervention accuracy, precision, and recall are low for both benchmark Intervention Problems and the Rush Hour Intervention Problem. The requirement of setting goal priors for Plan Recognition is an issue that must be overcome for intervention. This is because during execution, the user's plan may be subverted to achieve the undesirable state by environmental factors (such as attackers and hidden knowledge) regardless the priors. In Unsafe Suffix Intervention, we find that even when we assume reasonable goal priors, if the plans for u share long common action sequences with plans for d, Plan Recognition as Planning (PRP) approaches fail to correctly disambiguate between u and d. This affects the overall intervention accuracy by producing many false alarms and misses. In the intervention scenario we simulated with the Rush Hour domain, setting goal priors is even more problematic. This is because the user is informed about the presence of a forbidden vehicle and that the puzzle can be solved without moving it prior to executing the planning task. The result of that information being given to the user is that the goal prior for u is low compared to d. In our experiments, we show that the recognition accuracy for PRP approach drops when goal priors are set this way. We remove the dependency on goal priors by using features of projected remaining suffixes (in Unsafe Suffix Recognition) and the observed partial solution H (in Human-aware Intervention) to learn the differences between safe and unsafe plans from example training data and using the learned models to predict intervention.

We observed that different features (or their combinations) affect the intervention decision in the benchmark domains. Next, we describe which features were selected by the intervention recognition classifiers for the Intervention Graph and the Plan Space Sampling methods.

- **Blocks-1:** When Intervention Graph features are used, the decision tree, k-nearest neighbor and naive Bayes classifiers select distance to u to make the intervention decision. The logistic regression classifier selects distance to u and Risk to make the intervention decision. When using the Plan Space Sampling method, the decision tree, logistic regression, naive Bayes classifiers select the Landmark Completion Heuristic. The k-nearest neighbor classifier selects the median Causal Link Distance between the reference plan and the plans to u.

- **Blocks-2:** When Intervention Graph features are used, the decision tree classifier selects Risk to decide intervention. The decision tree, logistics regression and naive Bayes classifier use Risk, Desirability, distance to u and d, and the active attack landmark percentage collectively to decide intervention. When deciding intervention using the Plan Space Sampling method, the decision tree uses the median action set distance to u, minimum generalized edit distance for state sequences for u and median causal link distance to d features. The k-nearest neighbor, logistic regression and naive Bayes classifiers use all features in the Sampled Feature Vector.

- **EasyIPC:** When Intervention Graph features are used, the decision tree, k-nearest neighbor and naive Bayes classifiers select distance to u to make the intervention decision. The logistic regression classifier selects Risk to make the intervention decision. When deciding intervention using the Plan Space Sampling method, the decision tree and the naive Bayes classifiers use the Landmark Completion Heuristic. The k-nearest neighbor classifier selects the median Causal Link Distance between the reference plan and the plans to u. The logistic regression classifier uses several features: action set distance, causal link distance, state sequence distance, minimum remaining distance, minimum edit distance to u, and state sequence distance, minimum remaining distance to and minimum edit distance of action and state sequences for plans leading to d.

- **Ferry:** When using Intervention Graph features, all four classifiers select Risk to make the intervention decision. When deciding intervention with the Plan Space Sampling method, the decision tree classifier selects median causal link distance to u, median state sequence distance to u, minimum remaining distance to d and minimum edit distance to d as features. The k-nearest neighbor classifier selects the median state sequence distance and the minimum remaining distances to u to make the intervention decision. The logistic regression classifier uses the action set distance, causal link distance and state sequence distance, minimum edit distance in action and state sequences to u. In addition the classifier also uses the state sequence distance and minimum edit distance to d. The naive Bayes classifier uses action set distance and causal link distance to d, in addition to causal link distance to u, state sequence distance to u, edit distances for action and state sequences for u, and the Landmark Completion Heuristic.

- **Navigator:** When Intervention Graph features are used all four classifiers select Risk to make the intervention decision. When deciding intervention with the Plan Space Sampling method, the decision tree, k-nearest neighbor and naive Bayes classifiers use the Landmark Completion Heuristic. The logistic regression classifier use the action set distance, causal link distance, state sequence distance, minimum remaining distance to critical, minimum edit distance for state sequences and the Landmark Completion Heuristic.

We model intervention by recognizing the directly contributing actions using the Blocks-1, EasyIPC, Ferry and Navigator domains. For the Blocks-1 problems, when using the Intervention Graph feature vector, the observer can recognize when intervention is required by monitoring one or two features (e.g., Risk and remaining distance to u). A similar observation can be made when using the Plan Space Sampling feature vector (the Landmark Completion Heuristic, Causal Link Distance). The EasyIPC intervention problems also use a few features (e.g., the remaining distance to u and Risk) for deciding whether to intervene. The logistic regression classifier uses more features from the Plan Space Sampling feature vector compared to the other three classifiers to decide intervention for the EasyIPC domain. The intervention decision for the Ferry domain and the

Navigator domain rely only on the Risk feature when using the Intervention Graph feature vector. For the Ferry domain domain, intervention using the Plan Space Sampling feature vector require monitoring for many features. In contrast, for the Navigator domain three classifiers rely on the Landmark Completion Heuristic to decide intervention. The logistic regression classifier use many plan distance metrics in addition to the Landmark Completion Heuristic to make the intervention decision.

We use the Blocks-2 domain to model intervention by recognizing indirectly contributing sequences. For this case, more features from the Intervention Graph feature vector is required to make the intervention decision. The same can be observed when using the Plan Space Sampling feature vector.

Our findings comparing PRP and the learned model for Human-aware Intervention show that deciding to intervene based on plan cost differences (PRP) is not sufficient, especially when intervening human users. As seen from the solution length distributions in Section E.4, the longer the human user spends exploring the state space, the higher the likelihood that his partial plan will get closer to u. Therefore, we argue that intervention for human users require representations that capture characteristics of the actor's behavior in addition to the planning representations. The features based on actions and the features based on state extracted from H we propose for learning Human-aware Intervention capture the behavior patterns of the human user and as a result produce accurate learned models. However, a limitation of our proposed approach is that some features in the feature vector are domain-specific. Therefore, adopting the proposed approach in different planning domains may require feature engineering.

There are other methods one could use for generating $\mathcal{X}_\Diamond$, which we will explore in future work. In Unsafe Suffix Intervention, we generated $\mathcal{X}_\Diamond$ from the s_H using an automated planner without considering the observations in H. We then compared the suffixes in $\mathcal{X}_\Diamond$ to an observation compatible reference plan. (Ramírez & Geffner, 2009, 2010; Sohrabi, Riabov, & Udrea, 2016) propose methods to compile the observations into the domain theory, which allows the planner to find observation compatible plans. We can use this technique to also find observation compatible suffixes in $\mathcal{X}_\Diamond$. By making both the $\mathcal{X}_\Diamond$ and the reference plan compatible with the observations, we believe the plan distance features will be more accurate for the Plan Space Sampling method.

5.10 Closing Remarks

Intervention is a necessity for online assistive agents and safety critical decision making, where an observer determines how to guide a user toward a desirable outcome while avoiding undesirable outcomes. We propose Intervention as a solution to this problem and introduced two algorithms that combine automated planning and machine learning to decide whether or not the user's likely plan will avoid the undesirable state. Representing the user's task as a planning problem allows us to extract features of the user's plan space that can be used to produce learned models to recognize when intervention is required. Our first solution Unsafe Suffix Intervention, uses automated planning to project the remaining suffixes and extract features to differentiate between safe suffixes that avoid the undesirable outcome and unsafe suffixes that do not avoid the undesirable outcome. The second solution, Human-aware Intervention uses only the observed plan to extract features that can differentiate between safe and unsafe solutions. We showed that the two learning based intervention solutions dominate the state-of-the-art Plan Recognition algorithms in identifying when intervention is required.

In this work, our objective was to identify, given a sequence of observations, whether the user requires intervention while minimizing false positives and negatives. For our current implementation we assumed that the intervention comes in the form of a block or an alert message. The natural next step following intervention is helping the user decide what to do next. This extension is particularly important for observers where the user is a human user who would like to be guided towards the goal instead of being given the solution outright (e.g., an automated tutoring agent). We identify two sub-problems in helping the user decide what to do next. First, we can explore how automated planning can be used to gradually probe the search space of the remaining planning task following intervention. In certain cases, the user may want a quick, well-focused help. For example, in the Rush Hour puzzle, the observer can suggest the first move in the shortest remaining plan that avoids the forbidden vehicle as a hint after intervening. In other cases, the user would prefer more abstract suggestion. For example, in the Rush Hour puzzle, the observer can suggest the vehicles that must be moved to solve the puzzle (i.e., the landmarks of the planning task). In various stages of the puzzle solving task, the user may opt to use these suggestions differently. The second sub problem is explaining intervention and the follow-up to intervention. We can explore how effective different explanation models (e.g., contrastive, selective) are in explaining intervention and intervention Recovery to human users.

There are several other extensions to our current intervention framework, that we would like to explore as future work. In the planning domains we used to model intervention tasks, we assumed that the user's actions are deterministic and there is only one undesirable state that needs to be avoided. It is possible to relax these two assumptions and explore intervention in non-deterministic environments where the user needs to avoid multiple undesirable goals. However, in order for intervention to be meaningful, the intervention planning domains need to be descriptive enough to model complex tasks. We can explore the feasibility of adopting scenario building game environments like Minecraft for this purpose.

An interesting issue from our Rush Hour intervention study is how well the classifiers trained from user data would generalize to larger and more difficult Rush Hour puzzles. For example, the puzzle can be made difficult by adding more forbidden vehicles, more vehicles, and also introducing random exit points in the board. Addressing this question is left for future work.

Chapter 6

Interactive Human-aware Intervention

Using the Rush Hour domain, we study how an intervention model can be extended to allow intervention and help users complete cognitively engaging tasks by providing helpful hints. In the context of the Rush Hour puzzle solving task, a hint is a piece of information about the Rush Hour planning problem. We design helpful hints to allow the user to carefully probe the search space of the Rush Hour STRIPS planning task described in Section 5.7.2 in Chapter 5 while avoiding the undesirable state. In the Human-aware Intervention model described in Chapter 5, we assume that the user keeps presenting actions from plan he started off with and does not replan. In the Interactive Human-aware Intervention model, we relax this assumption. The intervening agent helps the user modify the current trajectory of the plan by providing the hints about the search space of the planning problem. Existing plan and goal recognition algorithms do not define a method for the user (or the software agent) to recover when the observer recognizes an undesirable state is developing and/or imminent. In this chapter, we address that limitation by designing four helpful hints the observer presents to the user after intervening.

- The minimum remaining number of moves
- The next best move
- The vehicles that must be moved
- Restart puzzle

In Chapter 5, we presented the Human-aware Intervention problem for the Rush Hour puzzle task as a tuple $\mathcal{I} = (D, \mathrm{d}, \mathrm{u}, H, o_i, \mathcal{X}_\Diamond)$ be a tuple where $D = \langle F, A, s_0 \rangle$ is a planning domain, $\mathrm{d} \subset F$ is a desirable state, $\mathrm{u} \subset F$ is an undesirable state, $H = (o_1[s_1], o_2[s_2], \ldots, o_{i-1}[s_H])$ is a history of previously observed actions and states, which started from s_0 with the implied resulting states in brackets. o_i is the *presented action* that the user would like to perform, and $\mathcal{X}_\Diamond = \emptyset$

because we can not use an automated planner to extract suffixes when a human user is solving the planning task. The Intervention Problem is a function $intervene(\mathcal{I}) : \mathcal{I} \rightarrow \{No, Yes\}$ that determines for the presented action o_i whether to intervene. For Rush Hour, the observer relies on those implied states instead of just the actions to decide intervention. Because we model the Rush Hour planning task as a deterministic state transition system, it is easy to map between actions and states.

For the study discussed in this chapter, the observer uses the Rush Hour Human-aware Intervention model configured to use the lowest level of sensitivity (i.e., $k = 1$). When $k = 1$, the observer predicts intervention one move before undesirable state occurs. For each action the user executes, the observer solves the Human-aware Intervention problem $\mathcal{I}$ if the $intervene(\mathcal{I}) : \mathcal{I} \rightarrow \{Yes\}$, the observer displays the hints to help the user decide what to do next. We evaluate the effectiveness of the proposed intervention recovery approach with a human subject experiment.

6.1 Motivation

When a human user is learning to use a new software application, it can be helpful to have a passive observer that can intervene to help the user reach his intended goal. This is particularly important when using the software imposes a significant cognitive load on the user and as a result the user's plan to accomplish the intended goal can become vulnerable to unintended actions or events. For this study, we simulate the human user learning to use a new software application with the Rush Hour puzzle solving task. In addition to the cognitive load, there may be information in the domain (e.g., Rush Hour puzzle, new software application) hidden to the user that may increase the likelihood that the user's intended goal will have unintended consequences. For example, at the start of the Rush Hour puzzle solving task, the user does not know where the forbidden vehicle is. Since the Rush Hour puzzles can be solved without moving the forbidden vehicle, moves that enable the forbidden vehicle to be moved later are *unhelpful moves* and signal to the observer that the human user needs help. Intuitively, any action that moves the forbidden vehicle triggers the

Table 6.1: Dimensions of the Human-aware Intervention Problem

Dimension	**Domain Specific Properties**
Actors in the environment	User, Observer
Goals hidden to the observer	User's goal not hidden One or more known undesirable states
Types of observations	The user's actions and states resulting from actions
Noise in observations	None
Intervention recovery	Offer helpful hint

undesirable state. The observer's decision making problem is to recognize when the user is making unhelpful moves before the undesirable state is triggered with varying sensitivity levels.

The rest of this chapter is organized as follows. We first revisit the dimensions of the Intervention Problem the observer solves during the Rush Hour puzzle solving task. As summarized in Table 6.1, this is a single-user Intervention problem, where no attackers, or competitors are present. Second we discuss related work on improving interactivity in human-agent collaborative environments, specifically focusing on Explainable Artificial Intelligence. Third, we formally define the four helpful hints. Fourth, we describe the experimental protocol and the software tools developed to support the experimental setup. Finally, we present the findings of our study to answer the following questions.

1. What are the most frequently used hints in different stages of the Rush Hour planning task?
2. Does the Interactive Human-aware Intervention have an effect on the solution length?
3. Does the Interactive Human-aware Intervention help move the user closer to the optimal solution?
4. Does seeing a help video along with the Interactive Human-aware Intervention affect the solution length?

6.2 Related Work

Developing human-aware intervention models must be predicated upon understanding how human users solve a planning task. The human subject study described in Chapter 5 addressed this requirement, where we used the plans generated by human users for Rush Hour planning tasks to generate learned models that can recognize when the user is making unhelpful moves and needs help. When the observer generates an intervention for the Rush Hour planning task (as an alert message displayed on the screen), it signals that an undesirable state is developing, which the user can not recognize on his own at the time. Although the intervention alert message may convey to the user that their action might cause undesirable consequences, given the cognitive load imposed by the task he/she may need help in framing the decision about what action to execute next. Studies have shown that users want some positive outcome from their action and are willing to accept that negative outcomes are a possibility (Good et al., 2005; Govani & Pashley, 2005; Debatin, Lovejoy, Horn, & Hughes, 2009; Byrne et al., 2016). Therefore, a solution cannot simply block the user from taking the action, but instead needs to assist them to decide what to do to recover from the unhelpful state. In this chapter, our objective is to study the impact of giving information about the Rush Hour planning problem on intervention recovery.

6.2.1 Improving Interactivity with Explanations

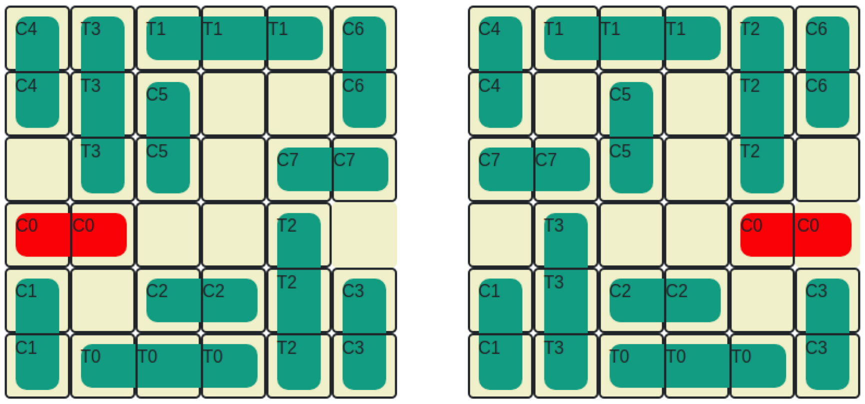
A human user and an agent collaborating in the same environment may have different knowledge (beliefs) about the environment. They may or may not know each other's own goals but will be expected to collaborate to achieve a common goal. Given a sufficiently complex task, these conditions may result in the agent executing some action that the human user does not expect, or vice versa. For example, in the Rush Hour domain, when the observer intervenes the human user, it may as a surprise because the user can not recognize the developing undesirable state. Recent research in Explainable Artificial Intelligence (XAI) addresses different challenges related to making AI systems transparent and interactive in human-agent collaborations. Specifically in the area of explaining the behavior of *rebellious agents* (Coman, Gillespie, & Muñoz-Avila, 2015), where an agent with goal reasoning capabilities may rebel by rejecting or revising the goals and plans a human teammate may expect it to follow, the explanation helps to build trust and understanding between the human and the agent. Rebellion has many positive motivations; for example in situations where the current plan execution will pose a threat to safety of the agent or the human, goals becoming unachievable due to resource constrains and ethical conflicts, differences in information access between the agent and the human. Similar to an agent who intervenes, a rebellious agent increases unpredictability when other agents or human users interact in the environment. In this situation, it is beneficial for the rebellious agent to have the ability to explain its behavior. In 2016, legal measures were adopted in the European Union that grant the individuals affected by automated decision making with a "right to explanation" (Parliament & Council, 2016).

Most efforts on improving interpretability of AI systems have focused on providing transparency to decision making using machine learning (Ribeiro, Singh, & Guestrin, 2016; Lundberg & Lee, 2017; Slack, Hilgard, Jia, Singh, & Lakkaraju, 2020). Research on improving interpretability and interactivity of AI systems that use automated planning for decision making has become popular in the recent years. Sohrabi et al. (2011) describe a technique to help humans understand a plan produced by an automated planner. Chakraborti et al. (2019) describe a method to generate explanations by reconciling the domain models of the agents and describing the differences between the models. Several other works focus on explaining why a planner chose a particular action rather than a different one (Langley, Meadows, Sridharan, & Choi, 2017; Fox, Long, & Magazzeni, 2017).

Prior efforts in explaining rebel behavior in agents consider explaining actions with harmful effects when the rebellious agent refuses to execute them (Briggs & Scheutz, 2015; Gregg-Smith & Mayol-Cuevas, 2015). Dannenhauer et al. (2018) proposes a method for a rebellious agent with goal reasoning capabilities to explain it's planning and goal reasoning decisions. When a goal reasoning rebellious agent rejects a goal, it needs to show that it can not find a plan to achieve that goal without violating certain conditions (e.g., safety, ethical etc.). The authors propose a question/answer dialog model where (1) the rebellious agent questions the current plans that violate the conditions and tries to find alternative plans or (2) the human questions the rebellious agent about the initial plan to find possible (safe) ways to achieve the goal. The question/answering dialog model uses the explainable planning framework proposed by Fox et al. (Fox et al., 2017). In that framework, the authors argue that questions in explainable planning must be designed to help the questioner *uncover a piece of important knowledge that the questioner does not have but believes to be available in the system* while avoiding the obvious statements. For an example, suppose our observer has the capability to explain intervention. When the user asks "why did you intervene?", the answer should not be "*because it will trigger the undesirable state in k number of steps*." This is because the response does not reveal any new information about the state of the world that caused intervention, or help the user learn how to avoid it. They present six questions: (1) "Why did you do action A", (2) "And why didn't you do something else (that I would have done)?", (3) Why is what you propose to do more efficient/safe/cheap than something else (that I would have done)?, (4) "Why can't you do that?", (5)"Why do I need to replan at this point?", (6) Why do I not need to replan at this point?" They also discuss what constitutes a response to these six questions from an automated planning perspective.

6.3 Hints: A Different Model of Explanation

The interactivity we bring in to Human-aware Intervention with helpful hints is different from the existing work on using explanations to improve how humans understand decision making of planning systems. In human-aware intervention, the human user acts as the planner but the complexity of the problem challenges the human user's ability to find a solution that avoids the undesirable state. The observer's role is to guide the user upon recognizing that the user is executing a plan that may result in the undesirable state. Following the definition of an explanation advanced by Fox et al. (2017), we designed the hints to help the user uncover pieces of information about the Rush Hour planning problem so that using the hints will help the user move toward the goal safely. Instead of engaging the human user in a question and answer dialog model, every time the



(A) Initial game state **(B) End game state**

Figure 6.1: A Rush Hour instance

observer intervenes, the hints are presented as options for the human user to choose. We designed the Rush Hour STRIPS planning task to generate the hints.

6.3.1 Cost Optimal Solutions in the (Modified) Rush Hour Planning Task

Let us revisit the Rush Hour instance introduced in Chapter 5. As illustrated in Figure 6.1, several cars (size=2) and trucks (size=3) are arranged on the board, which has only one exit. The objective of the game is to move the vehicles on the board in such a way that the target car (shown in red) can be moved out of the exit on the right edge of the board. Solution to the Rush Hour puzzle is a sequence of legal moves (a *plan*) that transforms the initial board state shown in Figure 6.1(A) to the goal state in Figure 6.1(B). Actions in the plan may have a cost that assigns a non-negative value to the action schema defined in the STRIPS domain definition. The cost function is given as $C : Op \rightarrow \mathbb{R}^0_+$. The cost of the plan $c(\pi)$ is $\sum(c(a_i))$. The optimal solution, the optimal plan π^*, minimizes the cost.

An off-the-shelf automated planner can be used on the Rush Hour STRIPS planning task defined in Section 5.7.2 in Chapter 5 to find a cost optimal plan. A given Rush Hour instance may have multiple cost optimal solutions. We recognize two optimizing criteria for a solution of the Rush Hour STRIPS planning task:

- the number of moves

- the number of vehicles moved

In this study, we consider the number of moves as the optimizing criteria where we assume all moves are of uniform cost equal to 1. For the puzzle shown in Figure 6.1(A), the cost optimal solution, minimizing the number of moves, consists of 21 moves. If optimized for the number of vehicles moved, the puzzle can be solved optimally by moving only 8 vehicles.

We modify the standard Rush Hour planning task by introducing a forbidden vehicle. For example, vehicle C2 in Figure 6.1(A) is forbidden. When the forbidden vehicle is moved, the undesirable state is triggered. The human user does not know what the specific forbidden moves are, but he knows that *some* moves are forbidden. The planning task can be solved without moving the forbidden vehicle. Therefore, if the human user moves the forbidden car while solving the Rush Hour planning task, it indicates that he is moving away from a cost optimal solution and needs help. Because moving the forbidden vehicle is unnecessary, an automated planner can be used to find cost optimal solutions to the Rush Hour planning task. In contrast, a human user not having the heurisitic search capabilities of the planner, may make mistakes and commit to plans that are longer and may even reach the undesirable state. For example, Table 6.2 shows two solutions to the same Rush Hour planning task illustrated in Figure 6.1. The cost optimal solution on the left is produced by an automated planner. The (partial) solution on the right is produced by a human user. The highlighted step in the human user's solution (18^{th} move) show that the user is moving the forbidden vehicle C2. In fact, the human user moved the forbidden vehicle twice before reaching the goal state. The human user's solution consisted of 43 moves. Furthermore, results from a human subject study discussed in Section 5.8.2 show that human users typically find longer plans to Rush Hour planning task compared to the cost optimal solutions found by an automated planner. Thus we conclude that the cost optimal solution to the modified Rush Hour planning task provides a reference solution against which we can compare the user's solution.

Table 6.2: Solutions for the Rush Hour planning task in Figure 6.1(A) produced by an automated planner (left) and a human user (right). The forbidden vehicle is C2

A safe, optimal solution (21 moves)	Unsafe solution (43 moves)
	MOVE-CAR C0 L17 L16 L18 WE
MOVE-CAR C0 L17 L16 L18 WE	MOVE-CAR C0 L16 L15 L17 WE
MOVE-CAR C0 L16 L15 L17 WE	MOVE-CAR C4 L30 L24 L36 NS
MOVE-TRUCK T3 L23 L17 L29 L35 NS	MOVE-CAR C4 L24 L18 L30 NS
MOVE-CAR C7 L20 L21 L19 EW	MOVE-TRUCK T3 L23 L17 L29 L35 NS
MOVE-CAR C6 L25 L19 L31 NS	MOVE-TRUCK T3 L17 L11 L23 L29 NS
MOVE-TRUCK T1 L32 L31 L33 L34 WE	MOVE-TRUCK T1 L34 L35 L33 L32 EW
MOVE-CAR C5 L28 L34 L22 SN	MOVE-TRUCK T1 L35 L36 L34 L33 EW
MOVE-TRUCK T3 L17 L11 L23 L29 NS	MOVE-CAR C7 L20 L21 L19 EW
MOVE-CAR C7 L21 L22 L20 EW	MOVE-CAR C6 L25 L19 L31 NS
MOVE-TRUCK T2 L14 L20 L8 L2 SN	MOVE-TRUCK T1 L34 L33 L35 L36 WE
MOVE-TRUCK T2 L20 L26 L14 L8 SN	MOVE-TRUCK T1 L33 L32 L34 L35 WE
MOVE-TRUCK T0 L3 L2 L4 L5 WE	MOVE-TRUCK T1 L32 L31 L33 L34 WE
MOVE-TRUCK T3 L11 L5 L17 L23 NS	MOVE-CAR C5 L28 L34 L22 SN
MOVE-CAR C7 L22 L23 L21 EW	MOVE-CAR C7 L21 L22 L20 EW
MOVE-CAR C7 L23 L24 L22 EW	MOVE-TRUCK T2 L14 L20 L8 L2 SN
MOVE-CAR C5 L28 L22 L34 NS	MOVE-TRUCK T2 L20 L26 L14 L8 SN
MOVE-TRUCK T1 L33 L34 L32 L31 EW	MOVE-CAR C2 L9 L8 L10 WE
MOVE-TRUCK T1 L34 L35 L33 L32 EW	MOVE-TRUCK T0 L3 L2 L4 L5 WE
MOVE-TRUCK T2 L26 L32 L20 L14 SN	MOVE-TRUCK T3 L11 L5 L17 L23 NS
MOVE-CAR C0 L15 L14 L16 WE	MOVE-TRUCK T3 L17 L23 L11 L5 SN
MOVE-CAR C0 L14 L13 L15 WE	MOVE-TRUCK T3 L23 L29 L17 L11 SN
	...

6.3.2 Hints: Formal Definitions

The observer solves the Human-aware Intervention problem ($\mathcal{I}$) for the Rush Hour planning task to recognize when the human user needs help and presents the hints. The model presented in Section 5.7 the observer automatically recognizes when a human user needs help during a cognitively engaging puzzle solving task. Hints are a special type of answer because they do not directly provide the complete solution to the question the user may ask the observer: "*why was I intervened?*" nor do they engage the user in a dialogue. Our design of helpful hints allows the user to carefully probe the search space of the Rush Hour planning task while avoiding the undesirable state. Hints are specially useful in situations where the human user wishes to retain some autonomy over the system he is interacting with (e.g., intelligent tutoring systems).

Formally, a hint is a function of the search space of the Rush Hour planning problem. We represent the observer's Rush Hour planning problem as a tuple $P_{observer} = (D_{observer}, \mathrm{u} \cup \mathrm{d})$ where $D_{observer} = (F_{observer}, A_{observer}, s_0)$ is the planning domain, $\mathrm{d} \subset F_{observer}$ is the desirable state, $\mathrm{u} \subset F_{observer}$ is the undesirable state. The observer also knows the history $H = (o_1[s_1], o_2[s_2], \ldots, o_i[s_H])$, which consists of the moves the human user made and the state resulting from each move. o_i is the last action before intervention and $[s_H]$ is the state resulting from that action. Note that we model the Rush Hour planning task as a deterministic problem. Therefore, the mapping between an action and the resulting state is straightforward. In order to generate H, the human user solves the Rush Hour planning problem $P_{user} = (D_{user}, \mathrm{d})$ where $D_{user} = (F_{user}, A_{user}, s_0)$ is the planning domain, $\mathrm{d} \subset F_{user}$ is the desirable state. Recall that u is hidden to the user.

In order to generate the hints, the observer first modifies $P_{observer}$ as follows:

- $F_{observer} = F_{user} \setminus \{$ `(car ?`C_i`)` or `(truck ?`T_i`)`$\}$, where C_i or T_i is the forbidden vehicle.

  $F_{observer} = F_{user} \setminus \{$ `(face ?`v_i `?`d_i`)` $\}$, where v_i is the forbidden vehicle

  $F_{observer} = F_{user} \setminus \{$ `(at ?`v_i `?`l_i`)` $\}$, where v_i is the forbidden vehicle

- $A_{observer} = A_{user}$
- $s_0 = s_H$
- $G = \{\mathrm{u} \cup \mathrm{d}\}$

The $P_{observer}$ is solved by an automated planner. For the modified $F_{observer}$, we remove the fluents (`face`, `at`, `car/truck`) associated with the forbidden vehicle from the fluent set F_{user}. This step instructs the planner to *forget* the existence of the forbidden vehicle. Although the forbidden vehicle is *removed* from the board, we block the cells previously occupied by the forbidden vehicle in F_{user} by not adding them to the fluents `free` in F_{user}. This step instructs the planner not to move the other vehicles to the *forgotten* forbidden vehicle's positions because those locations are not free. These two steps force the automated planner to find solutions that do not move the forbidden vehicle and ensures that the found plans do not violate rules of the game. The solution to $P_{observer}$ is a plan $\pi_{observer}$ from the $[s_H]$ until d is satisfied. We denote the cost optimal plan as $\pi^*_{observer}$.

Definition 12. *The hint is a tuple $\mathcal{N} = (F_{observer}, A_{observer}, [s_H], \mathrm{d}, \mathrm{u})$, where $F_{observer}$ is the modified Rush Hour planning domain defined above, $[s_H]$ is the state at the time intervention occurred as defined in H and d is the desirable goal and u is the undesirable state. The output of $\mathcal{N}$ is a property of the observer's planning problem $P_{observer}$.*

We study four properties of $P_{observer}$. The properties are derived considering the **cost optimal solutions** and the **fact landmarks** of $P_{observer}$.

- **The number of remaining moves**

  If $P_{observer}$ is solved optimally, what is the number of moves in the cost optimal plan $\pi^*_{observer}$ using the number of moves as the optimizing criteria.

- **The next best move**

  If $P_{observer}$ is solved optimally, what is the first move to execute in the cost optimal plan $\pi^*_{observer}$ using the number of moves as the optimizing criteria.

- **The vehicles that must be moved**

  We use the fact landmark definition advanced by Hoffmann et al. (2004) to find the vehicles that must be moved to solve $P_{observer}$. Fact landmarks are the fluents that must be true for every plan that solves $P_{observer}$. When the fluents are grounded, the set of vehicle objects associated with the fact landmarks are the vehicles that must be moved to solve $P_{observer}$.

- **Restart puzzle**

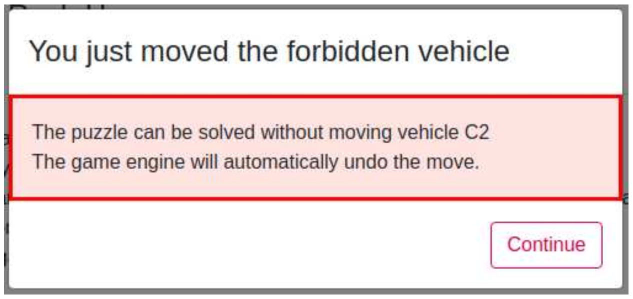
  Modifies $[s_H]$ to s_0 so that the user solves P_{user} again from the beginning.

- **Ignore hint**

  This option is not associated with properties of $P_{observer}$. It allows the user to disregard the hints and continue solving P_{user}. Two successive choices of *Ignore hint* disables the hints and stops the Human-aware Intervention process. If the user moves the forbidden vehicle, an alert is displayed to inform the user about the forbidden action and the move is automatically undone. Figure 6.2 shows the alert message. The automatic undo blocks the user from further searching down the undesirable path and forces the user to explore a different part of the search space for P_{user}.

6.4 Studying How Human users Solve the Rush Hour Planning Task

We implemented an online system, which allowed human users to solve a Rush Hour planning task with and without intervention. As the human user solves the planning task, the system records the events that occur on the interface. We use the system to conduct two rounds of experiments. In the first round, human users solved the Rush Hour planning task **without the Human-aware Intervention**. In Chapter 5, we used the event logs captured from this experiment as the observation history H to extract unique features that indicate when the user is getting too close to the undesirable state and needs help. Using the feature set, we generated learned models to predict whether or not the user's solution will reach an undesirable state before it actually happens. In the second



You just moved the forbidden vehicle

The puzzle can be solved without moving vehicle C2
The game engine will automatically undo the move.

Figure 6.2: Forbidden vehicle move alert message

round, we allow the human users to solve a Rush Hour planning task **with the Human-aware Intervention and the hints**. In this chapter we discuss the software framework we implemented for this purpose, the experiment protocol and the findings.

6.5 Rush Hour Puzzle Design Decisions

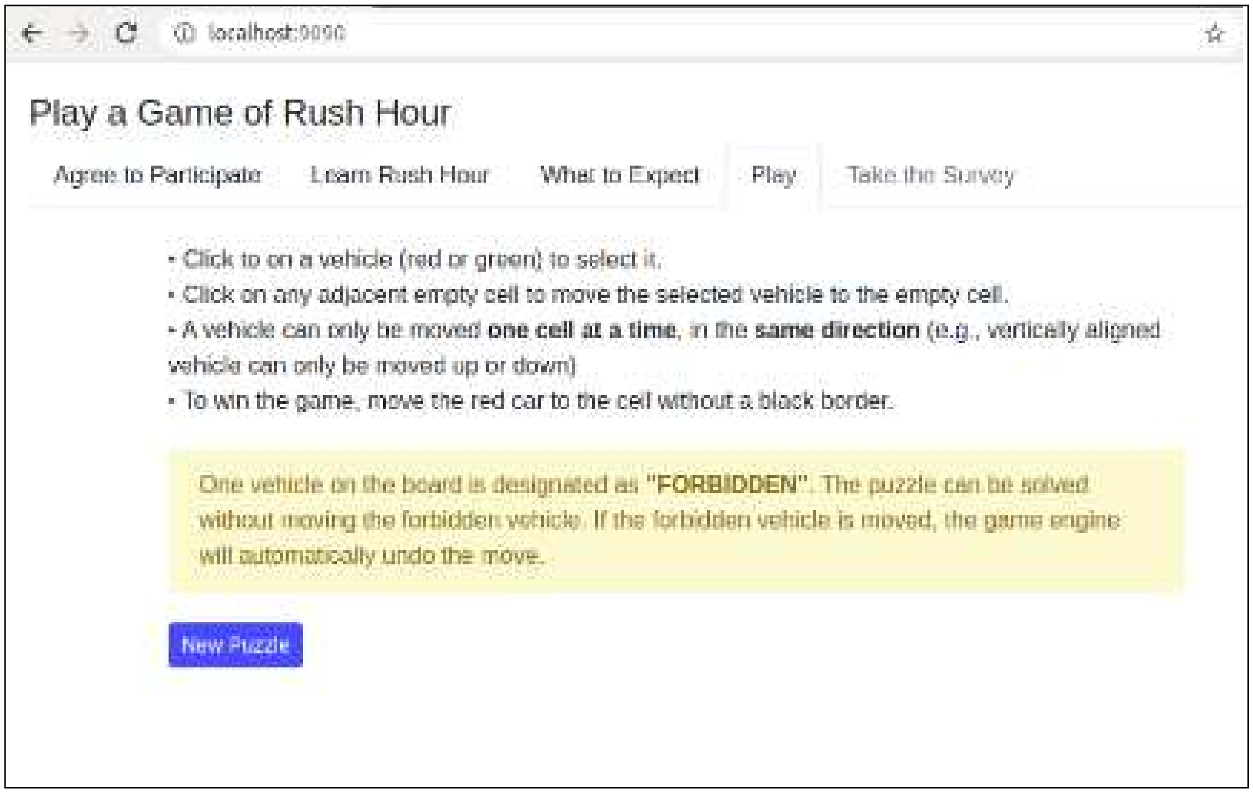
We use the Rush Hour puzzle as a supplementary task comparable to human users learning to use a new software or a web application. Therefore, the puzzle solving task should pose a sufficient challenge for the user during the search for a solution in order to make the intervention step more meaningful. Adding stationery vehicles (e.g., like the forbidden vehicle) is an alternative way to increase the difficulty of the puzzle without having to increase the number of vehicles on the board (Fernau, Hagerup, Nishimura, Ragde, & Reinhardt, 2003) to make the puzzle more challenging. In order to further instill the importance of avoiding the forbidden vehicle in the user's mind, we also provide warning messages on the Web interface (Figure 6.3) to inform the user about the presence of a forbidden vehicle and what would happen if the forbidden car was moved. The message also conveyed a *penalty* for moving the forbidden vehicles. When the user moves the forbidden vehicle, the move is automatically undone by the game engine. This penalty forces the user to think carefully about the next move to execute and prevents the user from searching further along the unhelpful parts of the search space. The penalty for moving the forbidden vehicle is a new addition compared to the conditions in the Rush Hour experiment described in Chapter 5

Recall that in Chapter 5, Section 5.8.1, we discussed three types of Rush Hour puzzles based on the position of the forbidden vehicle (**E**dge, **M**iddle, **C**orner). In the first round, where the human

Table 6.3: Rush Hour puzzles used for the experiment without Human-aware Intervention (left) and with Human-aware Intervention (right)

Type	Puzzles
C	P2, P4, P6, P8
E	P1, P3, P5, P9, P10
M	P7

Type	Puzzles
C	P2, P4, P6, P8
E	P1, P3, P5, P9, P10
M	P7, P11, P12, P13



- Click to on a vehicle (red or green) to select it.
- Click on any adjacent empty cell to move the selected vehicle to the empty cell.
- A vehicle can only be moved **one cell at a time**, in the **same direction** (e.g., vertically aligned vehicle can only be moved up or down)
- To win the game, move the red car to the cell without a black border.

One vehicle on the board is designated as **"FORBIDDEN"**. The puzzle can be solved without moving the forbidden vehicle. If the forbidden vehicle is moved, the game engine will automatically undo the move.

Figure 6.3: Message about the forbidden vehicle in the Rush Hour planning task

subjects solve the puzzle without Human-aware Intervention, we use 10 puzzles. The Human-aware Intervention learned models are developed using the data collected from the 10 puzzles. In the second round, where the human subjects solve the puzzle with Human-aware Intervention, we use 13 puzzles. Table 6.3 show the puzzle distribution. We added three extra puzzles to category M to resolve the distribution imbalance in the first round. We expect that these three classes of Rush Hour puzzles will pose unique challenges for the human user when trying to avoid the forbidden vehicle.

6.5.1 The Rush Hour Web Simulator

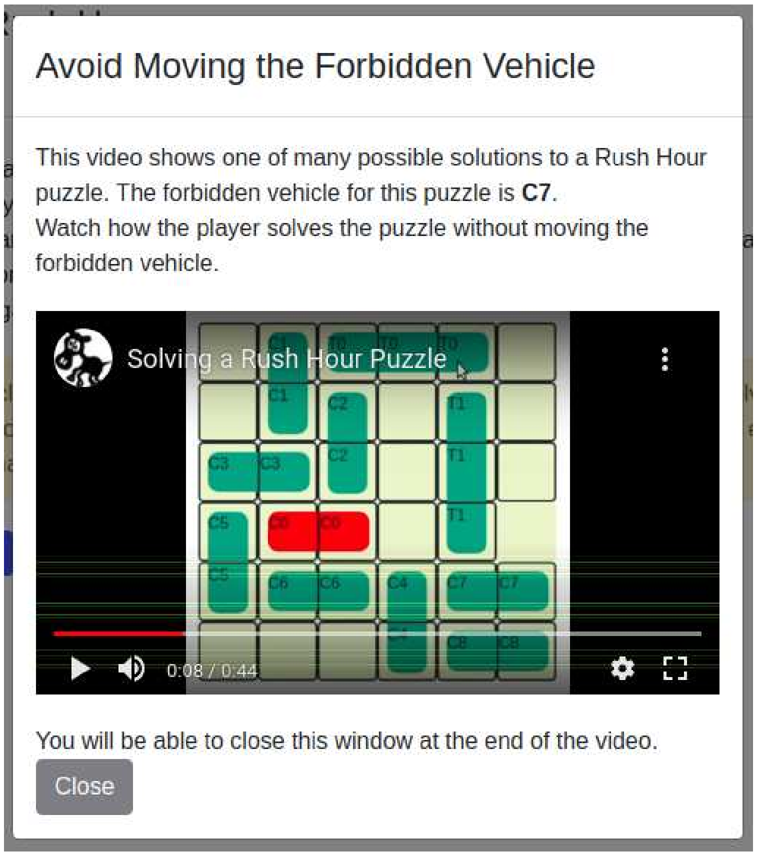
We designed the Rush Hour software framework to conduct human subject studies in an environment that allows the user to be engaged in a cognitively taxing task. The user interacts with several components of the Rush Hour Web Simulator:

- The consent agreement
- The Rush Hour tutorial
- The example solution video
- The Rush Hour solver (game board)
- The Post-study survey/debriefing

The components are displayed as tabs on the web page. If the Rush Hour planning task is solved with the Human-aware Intervention and the hints, then the human user can also view an example solution video that shows how to solve the puzzle without moving the forbidden vehicle. The example solution video is shown before the actual puzzle solving task begins. The Rush Hour solver component allows the user to start a puzzle solving task by clicking a button. Then a random Rush Hour puzzle is fetched from the server. The participant can move the vehicles on the board by clicking once on the vehicle to select it and then clicking once on an adjacent empty cell to move it to that cell. To comply with the classical planning model that uses discrete actions, the web application forces the user to move the vehicles one cell at a time. All illegal moves are blocked and the user is alerted if the he attempts to make an illegal move.

6.5.2 System Design Decisions

Our first design choice was in establishing the physical setup of the system. Because the experiments are targeted toward all kinds of computer software users (experts/non-experts), we wanted the subjects' attention to be mainly on the puzzle solving task and not be distracted by a complex user-interface. To allow for more users to participate, while minimizing the time commitment and



Avoid Moving the Forbidden Vehicle

This video shows one of many possible solutions to a Rush Hour puzzle. The forbidden vehicle for this puzzle is **C7**.
Watch how the player solves the puzzle without moving the forbidden vehicle.

You will be able to close this window at the end of the video.

Figure 6.4: Rush Hour example solution video

the effort, we wanted the system to be accessible from anywhere (e.g, home, university). Because of these reasons we designed the system to be a single page web application.

Our next design decision is concerned with capturing and storing experiment data. As per the university Institutional Review Board (IRB) regulations, the subject data is required to be stored in a secure manner. Further, because our system is deployed as a web application, multiple users may use the system at once and we need to ensure the data is transferred from the web application client to the server securely. We established communication between the client and the server over HTTPS via a stateless request-response scheme. The client side uses NodeJS, an asynchronous, event-driven JavaScript run-time, which is designed to build scalable network applications. The client also uses NodeJS security modules (Helmet) for improved security. The data is stored in a secure remote, server dedicated to the web application.

Our third design decision concerns with the interactivity of the system components. The design of the system needs to facilitate seamless transition between the two operating modes: learning behavior patterns that can be used as indicators of users' needing help and evaluating learned intervention models in situ. Therefore, we designed the web application to interact with micro-services via a RESTful API. Our system architecture is RESTful, in the sense that the client interacts with the server by sending HTTP GET and POST requests using JSON messages to an endpoint URL in the server. The API defines many services (persisting game state, intervention, providing hints etc.). These services are implemented as separate components on the server. REST API response payloads sent from the server are text and encoded in JSON. The response is parsed at the client and appropriate event is triggered on the user interface. In addition to supporting platform-independence, which encourages user participation, the services can be activated and de-activated based on the operational mode. This allows us to extend the existing system to evaluate how human users respond different intervention models in the future.

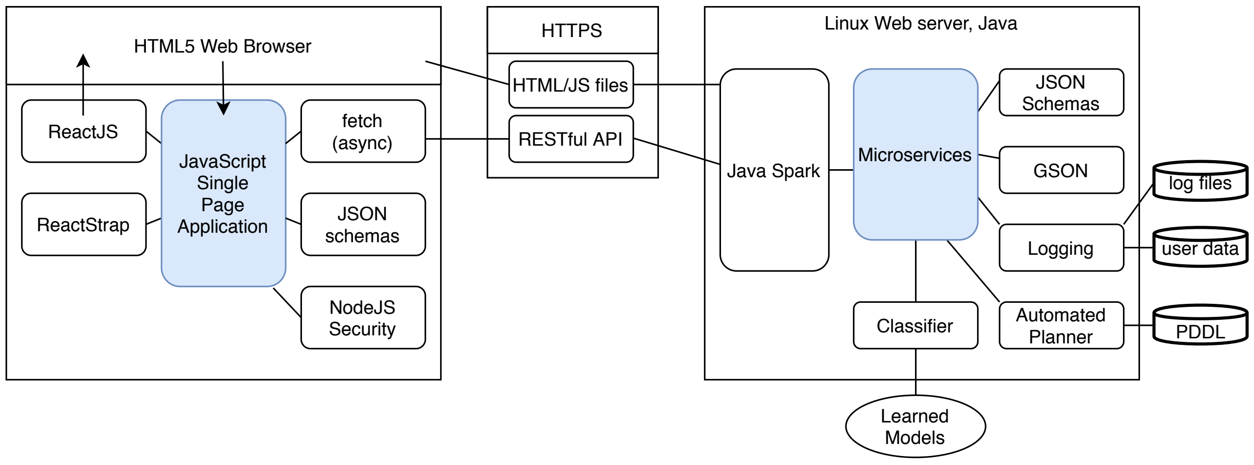


Figure 6.5: The client-server architecture of the Rush Hour Web Simulator

6.5.3 System Architecture

The basic system architecture is a JavaScript single page web application in an HTML5 browser that uses RESTful API to access micro-services provided by a Java server running on Linux. The client consists of a minimal `index.html` file that loads and executes the bundled JavaScript application. The client and server files are bundled into a single JAR file for the execution on the Linux server at a specified port. Figure 6.5 illustrates the system architecture.

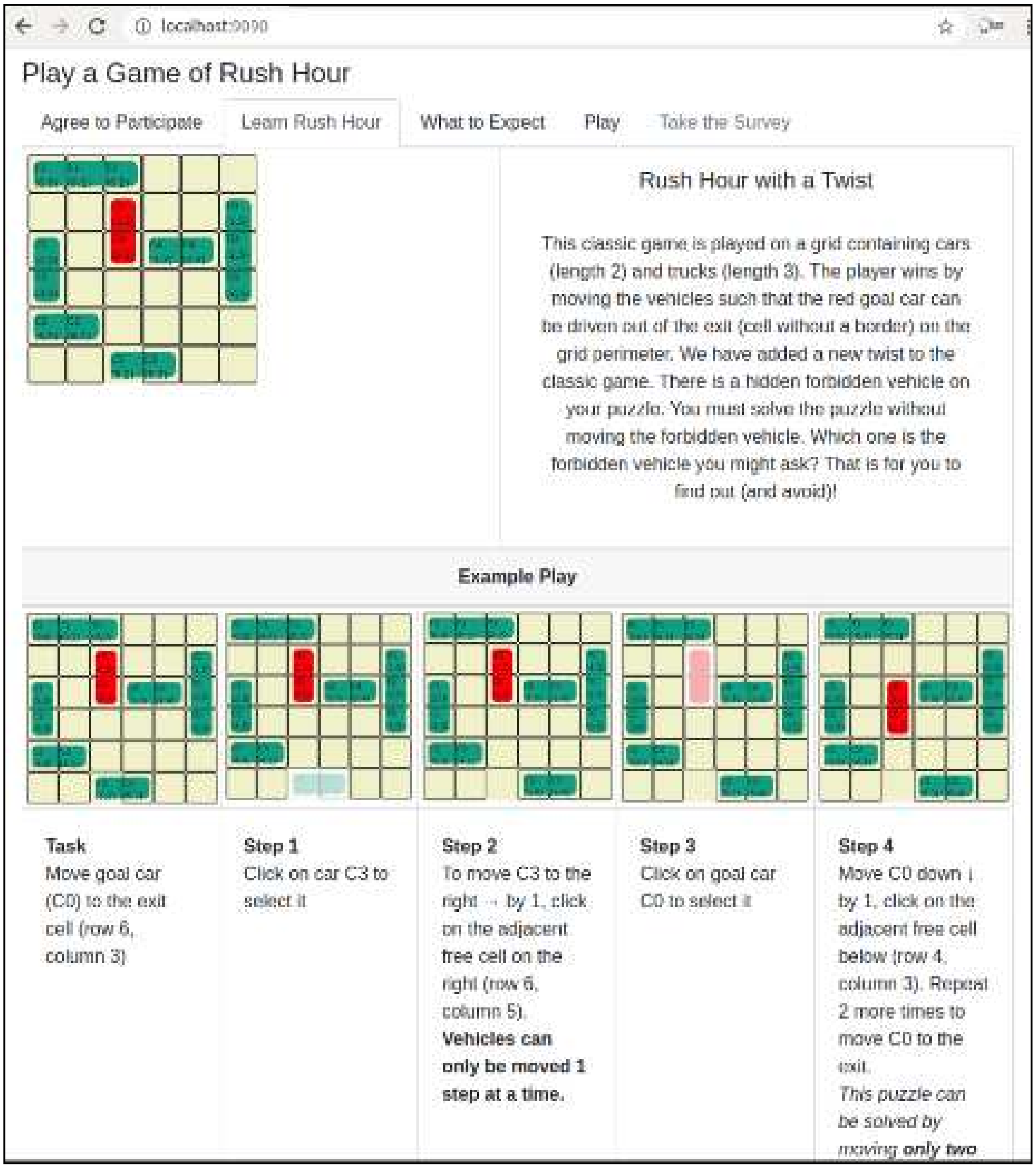
The system is a client-server application that communicates over HTTPS. On the client side, the browser loads the `index.html` file, which in turn loads the bundled JavaScript single page application `bundle.js`. The single page application makes RESTful API requests to the server on the same port using JavaScript's asynchronous fetch. We defined several JSON schema to enable message passing between the client and the server via the RESTful API. The JSON schema verify requests on the server side and responses on the client side. Any verification failures are handled by error responses, also defined in the JSON schema. ReactJS renders the application using ReactStrap on the client. Because the communication between the client and the server occurs over the Internet, we use the NodeJS security module Helmet to secure the HTTP headers to prevent attacks like cross-site scripting.

On the server side, GSON is used to convert the JSON requests from the client to Java objects and Java objects to JSON responses. The server uses Java logging to record user data (moves, survey responses on the client side) and errors that occur on the server side on text files. The component that contains the learned model, which implements Human-aware Intervention and the automated planner components are activated only when the system is used to evaluate learned intervention models. The automated planner component uses PDDL problem and domain definitions to retrieve information about $P_{observer}$ to help guide the user using the hints.

As the subject interacts with each component, unique API calls are triggered to inform the server about the status of the game. The consent agreement component lets the subject read the terms and conditions of the study and give informed consent to participate in the study as required by the IRB. When the user gives consent to participate in the study, the user gets assigned a unique identifier and components 2 and 3 get activated. This unique identifier is maintained at both the client side (as the Local Storage entry in the Web browser) and on a text file at the server until the user's session ends. The Rush Hour tutorial presents a brief introduction to the Rush Hour puzzle, the rules and the objective of the puzzle. Figure 6.6 shows the tutorial for the Rush Hour planning task. It also shows an example play to educate the user on how to select and move objects on the web interface. The consent agreement and the tutorial are static components, in the sense the user does not have a lot of flexibility to change the contents through interaction. The example solution video allows minimal interaction to pause and replay the solution video. The video component is only available if the Rush Hour Simulator is operating with the Human-aware Intervention. On the other hand, the solver is a dynamic and an interactive component. The solver component allows the user to start a puzzle solving task by clicking a button, which fetches a random Rush Hour puzzle from the server. Figure 6.7 shows how the puzzle is displayed to the human user. The participant can move the vehicles on the grid by first clicking once on the vehicle to select it and then clicking once on an adjacent empty cell to move it to the selected cell. We enforced the discrete actions in the environment by invalidating the moves that jump over multiple cells on the puzzle boards. The other illegal move is user attempting to move a vehicle in the opposite orientation. When the user makes one of these illegal moves, error messages are displayed and the move is blocked. All legal moves the user make are temporarily kept in in-memory data structures on the client side. Once the user finishes the puzzle solving task, an API call is invoked to initiate the data transmission between the client and the server in a secure manner via HTTPS. The server records the users solution on a text file in a directory created with the user's identifier. When the user's solution is stored on the server successfully, the post-study survey and debriefing component is activated. If the Rush Hour Web Simulator operates without the Human-aware Intervention mode, the survey collects demographic data and the user's general puzzle solving habits. If the Rush Hour Web Simulator operates in the Human-aware Intervention mode, the survey collects participant demographics and a subjective rating on how useful the hints were to the user. Once the user completes the survey, the data is transmitted to the server and stored as text files.

6.5.4 The Rush Hour Web Simulator - Client-side Operation

Figure 6.8 illustrates the component structure of the Rush Hour Web Simulator client. The **WebApp** component integrates the sub components: the consent agreement, the Rush Hour tutorial, the Rush Hour solver, the user help and the survey. The user help component consists of interactive Human-aware Intervention messaging model (implemented in the **Intervention** component) and the help video (implemented in the **Video** component). The Rush Hour solver (implemented in the **Board** component) is further broken down in to sub components, which model different objects in the game such as cars, trucks and squares. The **Game** component implements the rules of the Rush Hour puzzle. The **Survey** component contains two components that implement the debriefing statement and the post study questionnaire. Each client component communicates using API calls within them. The WebApp component invokes the API calls on the sever component in the Rush Hour Web Simulator to fetch intervention decisions and log the events that take place on the Web Simulator interface. When the Human-aware Intervention occurs the Intervention component displays a pop-up message with the hints (Figure 6.9 - left). When the user selects a hint, the client makes another API call to fetch the appropriate property value of the $P_{observer}$ as the response from the server. The response is displayed as a follow up dialog box and the user is allowed to continue on with the Rush Hour planning task (Figure 6.9 - right). Figure 6.9 shows the intervention message



Play a Game of Rush Hour

Rush Hour with a Twist

This classic game is played on a grid containing cars (length 2) and trucks (length 3). The player wins by moving the vehicles such that the red goal car can be driven out of the exit (cell without a border) on the grid perimeter. We have added a new twist to the classic game. There is a hidden forbidden vehicle on your puzzle. You must solve the puzzle without moving the forbidden vehicle. Which one is the forbidden vehicle you might ask? That is for you to find out (and avoid)!

Example Play

Task
Move goal car (C0) to the exit cell (row 6, column 3)

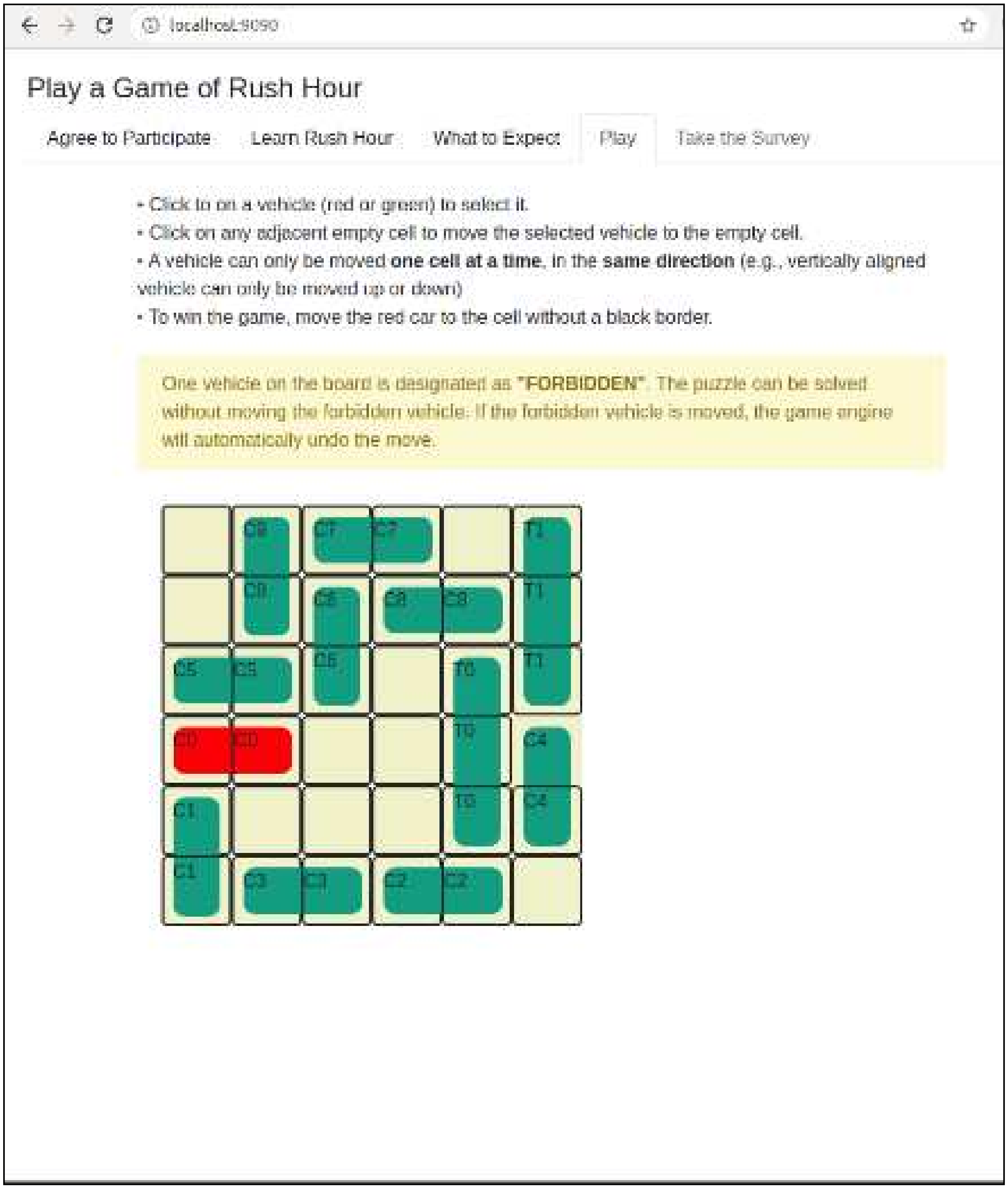
Step 1
Click on car C3 to select it

Step 2
To move C3 to the right → by 1, click on the adjacent free cell on the right (row 6, column 5). **Vehicles can only be moved 1 step at a time.**

Step 3
Click on goal car C0 to select it

Step 4
Move C0 down ↓ by 1, click on the adjacent free cell below (row 4, column 3). Repeat 2 more times to move C0 to the exit.
This puzzle can be solved by moving ***only two***

Figure 6.6: The Rush Hour planning task tutorial



localhost:9090

Play a Game of Rush Hour

Agree to Participate | Learn Rush Hour | What to Expect | Play | Take the Survey

- Click to on a vehicle (red or green) to select it.
- Click on any adjacent empty cell to move the selected vehicle to the empty cell.
- A vehicle can only be moved **one cell at a time**, in the **same direction** (e.g., vertically aligned vehicle can only be moved up or down)
- To win the game, move the red car to the cell without a black border.

One vehicle on the board is designated as **"FORBIDDEN"**. The puzzle can be solved without moving the forbidden vehicle. If the forbidden vehicle is moved, the game engine will automatically undo the move.

Figure 6.7: The Rush Hour planning task solver interface

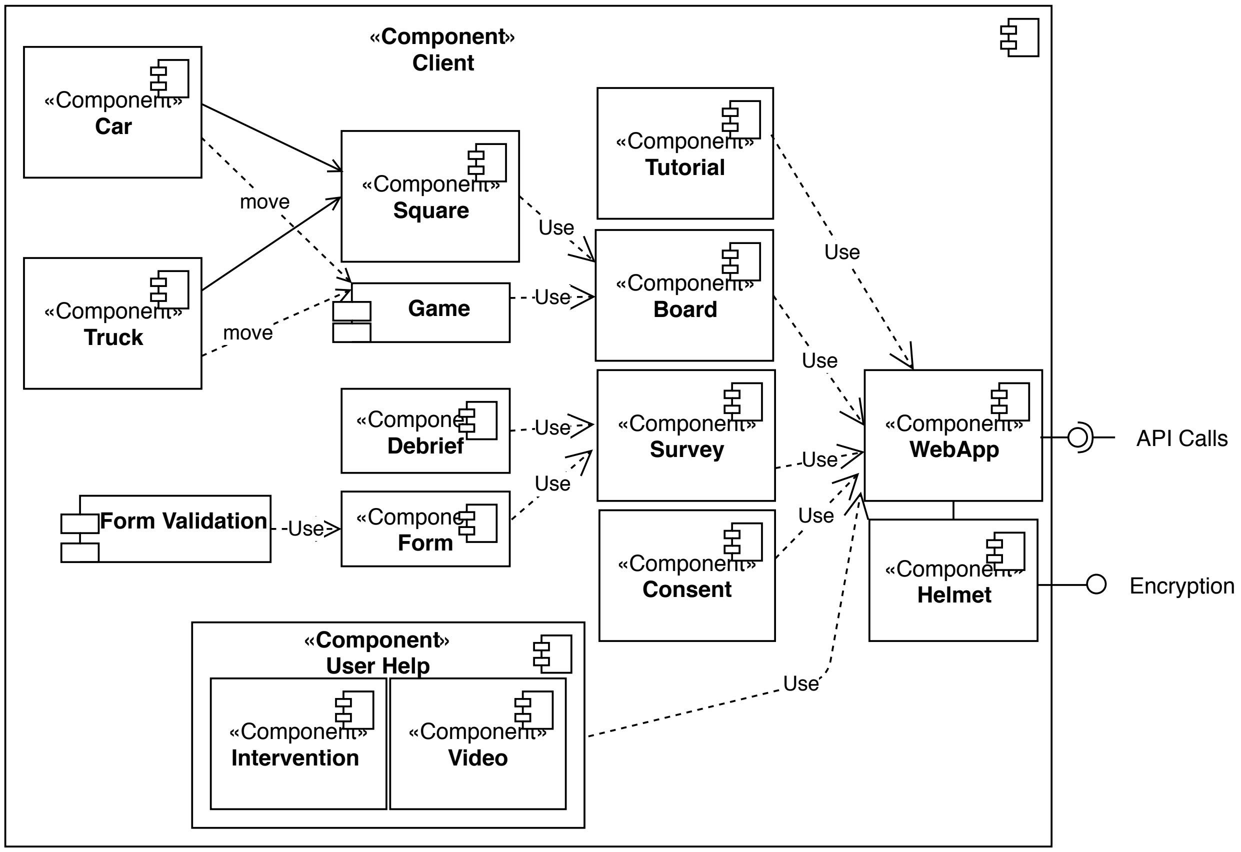
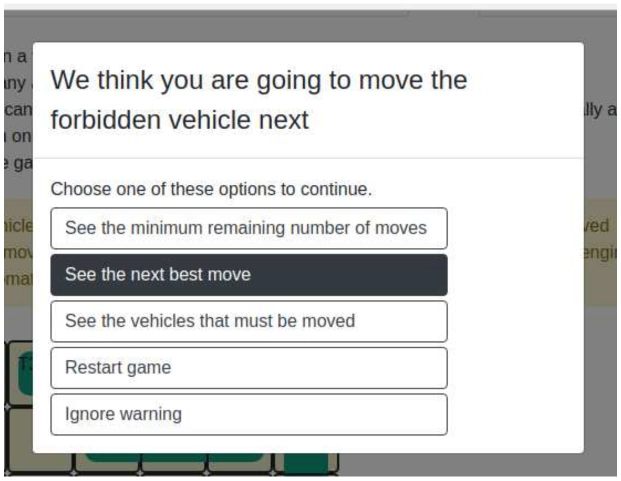


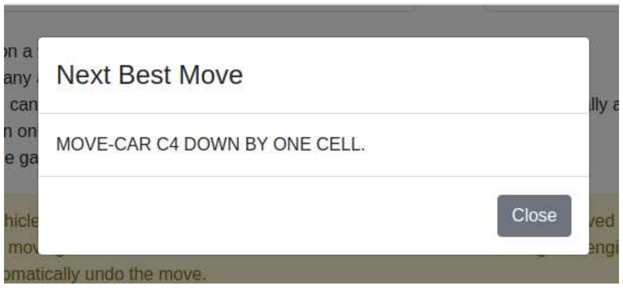
Figure 6.8: UML component diagram for the Rush Hour Web Simulator client



We think you are going to move the forbidden vehicle next

Choose one of these options to continue.

See the minimum remaining number of moves
See the next best move
See the vehicles that must be moved
Restart game
Ignore warning



Next Best Move

MOVE-CAR C4 DOWN BY ONE CELL.

Figure 6.9: Interactive Human-aware Intervention (left) and the hint response (right)

shown to the user (left) and the user requesting to see the next best move as the hint. Subsequently, the server responds with the property value of the $P_{observer}$ and the client displays the message on the right. One of the hint options is to let the user ignore Human-aware Intervention. The Intervention component keeps track of how many times the user has chosen to ignore intervention and after two consecutive ignore requests, the client terminates fetching hint responses from the server. When the Rush Hour planning task finishes, the user's solution and the Human-aware Intervention decisions received during the planning task are automatically transmitted to the server to be saved as text files.

6.5.5 The Rush Hour Web Simulator - Server-side Operation

As shown in Figure 6.10, the Rush Hour Web Simulator server responds to client's API calls through the **MicroServer** component. The MicroServer consists of messaging frameworks (**GSON**, **Spark**, **JSON**) and message logging (implemented in the **Logger** component). In addition, the **Data Persistence** component supports saving the user solutions, client-side events in text files. The **Intervention** component and the **Planning** component on the server are used to recognize Human-aware Intervention and generate helpful hints respectively. The **Intervention** component uses a learned Human-aware Intervention model to identify when the user needs help. The **Planning** component generates the responses to the user's hint requests (see Definition 12) by solving $P_{observer}$ using an automated planner. We integrated the Fast Downward planner (Helmert, 2006) to the Planning component to generate cost optimal plan for $P_{observer}$. The Fast Downward planner was configured to use the A-star search algorithm with the admissible heuristic Landmark-cut (`lmcut`) (Helmert & Domshlak, 2009) to find cost optimal plans.

We use a minimalist and responsive design (supported by `ReactStrap` library) for the Rush Hour Web Simulator interface to help the user to start the puzzle solving task with a very small learning curve. The simple, non-intrusive messaging model we designed for the client-server communication automatically sends the event data back and forth between the client and the server and helps the user focus completely on the puzzle solving task. The system is deployed on a secure

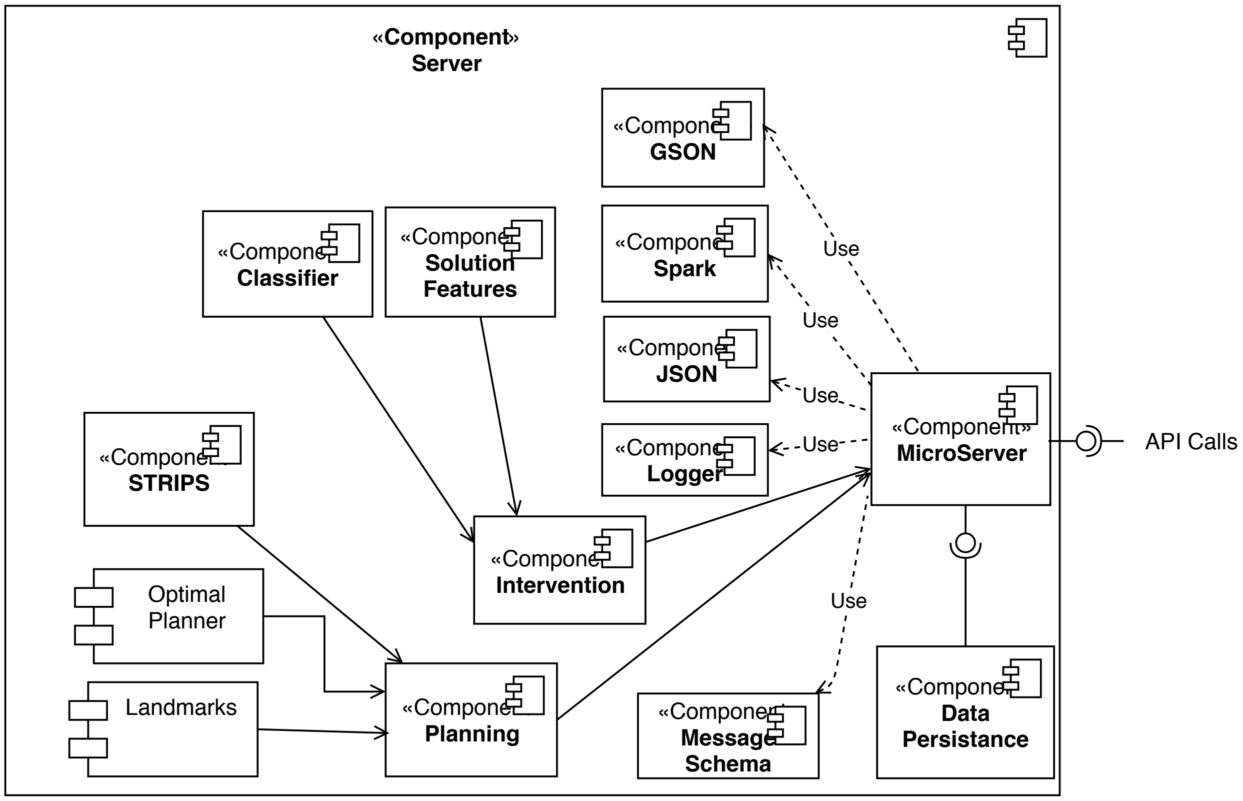


Figure 6.10: UML component diagram for the Rush Hour Web Simulator server

Linux Web server with a public IP address, which allows the users to access the application from anywhere on a variety of devices.

6.6 Experiment Design

In the first round of data collection, the human users solved Rush Hour planning tasks without Human-aware Intervention (see Appendix E.2 for details). In the second phase, the application intervenes the user and provides helpful hints. We refer to the first group as the **control group**. We discussed the findings of this study in Chapter 5. In this chapter, we discuss the findings of the second experimental condition where the human users solved the Rush Hour planning tasks with hints. We refer to the second group as the **condition group**. Therefore, we have a between group experimental design to evaluate the effect of using the Interactive Human-aware Intervention between the control and the condition group.

For the condition group, we recruited 142 participants from a university student population. The participants were not compensated for their time. Upon agreeing to participate in the study each subject is assigned one puzzle randomly out of 13 puzzles. In addition, the participants are randomly selected to watch a 44 second video of a sample game play. The video showed how a Rush Hour planning task can be solved by avoiding the forbidden vehicle. All participants had the option to use the only tutorial available on the Rush Hour Web Simulator to learn the rules of the puzzle. The participants were informed that one of the vehicles on the board is forbidden and the puzzle must be solved without moving it. During the Rush Hour planning task, the participant is shown the hints when the learned model recognizes that the he/she needs help (see Figure 6.9). The participant can follow a hint or decide to ignore it. Based on the user's choice, a second alert message is shown, with the corresponding property of $P_{observer}$. Once the puzzle solving task is completed, the participants are asked to complete a short survey to capture the demographics and also asks them to subjectively rate the helpfulness of each hint type on a 5-point Likert scale. Figure 6.11 illustrates the activity sequence of the experiment.

6.6.1 Collecting Data

From the 142 participants, we removed records of 7 subjects because they had quit the session before the Rush Hour planning task was completed. The remaining 135 records are used in the evaluation. 130 of the 135 participants also completed the post-study survey. For each participant, we record the solution, requested hints, the demographic information and the 5-point helpfulness rating for the hints.

6.7 Results and Evaluation

We first present the findings of the post-study survey, discuss the properties of our study sample and outline how human users subjectively rated the helpfulness of the hints. Next, we turn our attention to the solutions produced by human users under two experimental conditions. We present the summary statistics for the number of moves in the solutions in the control and the condition

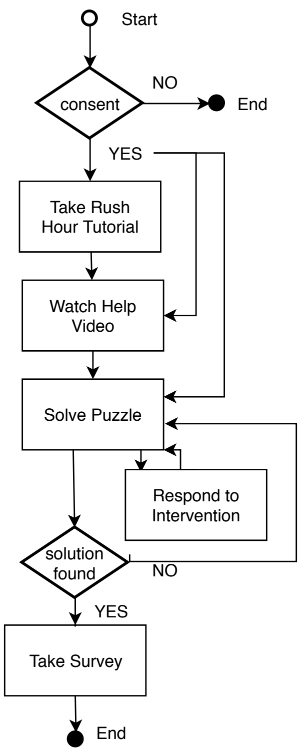


Figure 6.11: Activity sequence for evaluating hints during intervention for the Rush Hour human subject experiment

groups and compare the solutions that moved the forbidden vehicle and the solutions that did not. We use this analysis to statistically verify if there is any relationship between avoiding the forbidden vehicle and the number of moves for both the condition and the control groups. Next, we discuss the usage of hints starting with subjective ratings for the helpfulness of hints given by human users. Next, we extend the discussion to study which hints were requested the most for different planning tasks. We also analyze the hint usage frequency during different stages of the planning task to identify if there are any patterns for the type of hint requested at early stages of the planning task compared to the late stages. Next, we evaluate the effectiveness of the Interactive Human-aware Intervention using four evaluation metrics.

6.7.1 Findings of the Post-study Survey

From the 131 participants who completed the demographics survey, the majority (52) were between the age of 20 - 25. There were 51 participants whose age was less than 20 years. Addi-

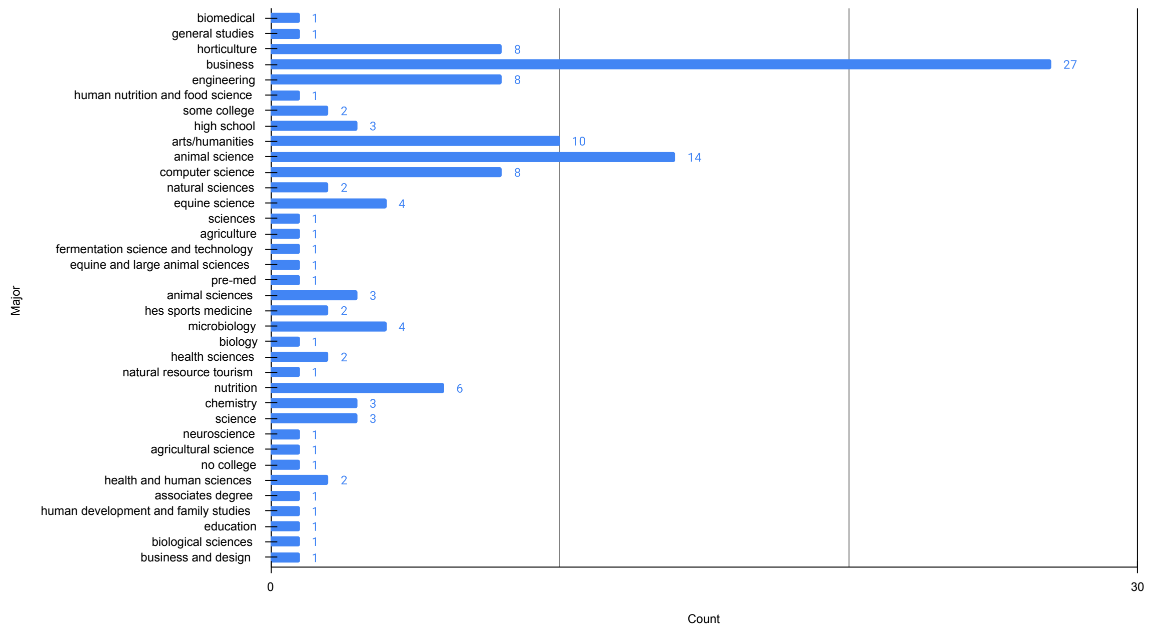


Figure 6.12: Educational backgrounds of the study participants

tionally, there were 12 participants each from 31 - 25 age group and 26 - 30 age group. There was 1 participant in the 36 - 40 age group. Two participants were above the age of 41. We were able to recruit participants from different educational backgrounds. As illustrated in Figure 6.12, the majority of the participants (27) are Business majors, followed by Animal Sciences (14) and Arts and Humanities (10). 47% of the participants have seen the Rush Hour puzzle before.

From the 135 participants who completed the Rush Hour planning task, 44 subjects did not use any hints. From the remaining 91 participants, 4 subjects did not complete the demographics survey. So we were unable to collect demographic data on them. We also asked the participants to rate the hints in 5 point rating scale (1 being the lowest rating and 5 being the highest) based on how helpful the hints were in helping them avoid the forbidden vehicle. Table 6.4 summarizes the mean and the standard deviation of the subjective ratings. It can be seen that across all three puzzle types, the hint "**Show the next best move**" has the highest subjective helpfulness rating. Considering the puzzle types C, E and M, we could not find unique subsets of hints the participants preferred the most.

Table 6.4: Mean (standard deviation) of the subjective helpfulness rating the subjects assigned for the hints **H1**: Show the minimum remaining number of moves, **H2**: Show the next best move, **H3**: Show the vehicles that must be moved, **H4**: restart puzzle

			Mean (std. dev.) helpfulness rating			
Type	**Puzzle ID**	**Count**	H1	H2	H3	H4
C	P2	6	1.5 (2.5)	**4.3** (2.6)	3.8 (2.3)	2.0 (2.3)
	P4	11	1.3 (2.1)	**2.3** (2.1)	1.5 (2.2)	0.8 (1.7)
	P6	3	2.0 (1.2)	**3.3** (2.6)	1.7 (2.9)	0
	P8	1	0	**3.0**	**3.0**	1.0
E	P1	4	1.5 (1.3)	**4.3** (1.0)	3.8 (1.9)	2.0 (2.2)
	P3	13	1.3 (1.6)	**2.3** (2.2)	1.5 (1.7)	0.8 (0.9)
	P5	13	1.4 (1.3)	**2.2** (1.9)	**2.2** (2.0)	1.8 (1.9)
	P9	7	0.9 (1.2)	**1.7** (2.2)	1.1 (1.7)	1.3 (2.0)
	P10	8	0.9 (1.0)	**2.8** (2.3)	1.6 (1.5)	**2.8** (2.1)
M	P7	6	2.0 (2.1)	**3.5**(2.1)	2.5 (1.9)	1.8 (1.3)
	P11	3	2.0 (2.6)	**3.3** (2.9)	2.0 (2.6)	**3.3** (2.9)
	P12	1	1.0	**5.0**	**5.0**	1.0
	P13	11	1.2 (1.8)	**1.7** (1.8)	0.9 (1.4)	0.9 (1.6)

6.7.2 Safe and Unsafe Solutions in the Control and the Condition Groups

We now compare the solution lengths of the control and the condition groups by breaking each group's solutions into two types: safe and unsafe. We refer to solutions that did not move the forbidden vehicle as **safe** and solutions that moved the forbidden vehicle as **unsafe**. In the control group, 66 users from the total 136 produced solutions that involved moving the forbidden vehicle (49%). From those who moved the forbidden vehicle, 54 users moved the vehicle more than once (82%). Table 6.5 describes the summary statistics for the solutions that did not move the forbidden vehicle and the solutions that moved the forbidden vehicle produced by the human users in the control group.

In the condition group 44 subjects did not use any hints. This is because the Human-aware Intervention agent did not classify the partial solutions they were producing as requiring intervention. Therefore, we only use the data from 91 subjects who solved Rush Hour planning tasks with hints for the analysis. Table 6.6 describes the summary statistics for the solutions in the condition group. 65 users from the sample of 91 solved the planning task but failed to avoid the forbidden vehicle (71%). In general, where there are a set of safe and unsafe solutions for a given planning

Table 6.5: Frequency, minimum, maximum, mean and std. deviation number of moves in users' solutions for the Rush Hour planning tasks P1 through P10 **solved without using the Interactive Human-aware Intervention**

PID	Safe Solutions					Unsafe Solutions				
	Freq	Min	Max	Mean	Std. dev.	Freq	Min	Max	Mean	Std. dev.
P1	18	24	106	43.9	20.5	-	-	-	-	-
P2	3	44	158	99.3	57.1	8	78	378	190.3	120.0
P3	-	-	-	-	-	12	25	50	35.5	8.3
P4	9	23	46	30	7.1	7	25	124	67.4	33.9
P5	4	23	32	26.5	3.9	7	14	82	32.0	23.3
P6	14	22	55	29	10.3	-	-	-	-	-
P7	2	29	37	33	5.7	9	43	132	80.9	38.2
P8	18	9	12	9.3	0.8	-	-	-	-	-
P9	2	21	27	24	4.2	14	29	169	66.3	39.0
P10	-	-	-	-	-	9	44	158	81.2	39.2

task, it can be seen that the mean solution length for the unsafe set is longer than the mean of the unsafe set. This indicates that the longer the human user spends searching for the solution, the more likely he will move the forbidden vehicle. There are several similarities between the solution lengths for planning tasks in the control and the condition groups. First, the planning task P1 did not produce any unsafe solutions for both groups. The reason for this observation is that when examining the structure of P1 it can be seen that moving the forbidden vehicle makes the planning task unsolvable. Second, the planning task P8 did not produce any unsafe solutions for both groups. Furthermore, P8 was the only planning task in type C planning task that did not produce any unsafe solutions. Third, in both groups, human users found it difficult to avoid the forbidden car for type E planning tasks, where the forbidden car was positioned along the edge of the board. For type E planning tasks (P3, P5, P9, P10), there are more unsafe solutions than safe solutions and also the mean solution length for unsafe solutions is larger compared to the safe solutions mean. We did not find a common pattern for the solutions generated by type M planning tasks. We only had 1 planning task for type M (P7) in the control group. We had four planning tasks of the same type (P7, P11, P12, P13) in the condition group. The users in the control group difficult to solve P7 without moving the forbidden vehicle. In contrast, the solutions generated by the users in the condition group for P7 were evenly split between the safe and unsafe groups. The same can

Table 6.6: Frequency, minimum, maximum, mean and std. deviation number of moves in users' solutions for the Rush Hour planning tasks P1 through P13 **solved using the Interactive Human-aware Intervention**

PID	Safe Solutions					Unsafe Solutions				
	Freq.	Min	Max	Mean	Std. dev.	Freq.	Min	Max	Mean	Std. dev
P1	4	51	80	59.7	13.7	-	-	-	-	-
P2	-	-	-	-	-	8	67	419	157.1	114.3
P3	1	51	51	51	-	12	43	121	71.8	24.8
P4	2	34	78	56	31.1	9	27	87	55.1	19.7
P5	5	27	49	37.6	9.5	9	34	154	54.9	37.7
P6	2	34	45	39.5	7.8	1	62	62	62	-
P7	3	27	59	39	17.4	3	44	195	98.7	83.7
P8	1	14	14	14	-	-	-	-	-	-
P9	-	-	-	-	-	7	22	71	43.7	19.2
P10	-	-	-	-	-	8	46	74	58.4	9.8
P11	1	61	61	61	-	2	50	95	72.5	31.8
P12	1	41	41	41	-	-	-	-	-	-
P13	6	24	40	29.5	6.3	6	30	55	41.2	9.6

be observed for puzzle P13. Figure 6.13 illustrates the percentage split between unsafe and safe solutions for the control group. Figure 6.14 illustrates the percentage split for the condition group.

6.7.3 Solution Lengths Compared to the Cost Optimal Solution in the Control and the Condition Groups

In Section 6.3.1, we show evidence from our previous study (Appendix E) to support the claim that the cost optimal solution to the modified Rush Hour planning task provides a reference solution against which we can compare the user's solution. We adopt the same principle to discuss the solutions the human users produced in the Human-aware Rush Hour planning task with hints. In both cases we use an automated planner to find the cost optimal solutions to the Rush Hour planning tasks.

We use 13 Rush Hour planning tasks in this study. Table 6.7 shows the optimal costs for the Rush Hour planning tasks $P_{observer}$ and P_{user}, using the optimizing criteria, the **number of moves** and the **number of cars**. Recall that $P_{observer}$ forces the planner to find solutions without using the forbidden vehicle. In contrast, P_{user} can find shorter solutions while using the forbidden vehicle. In

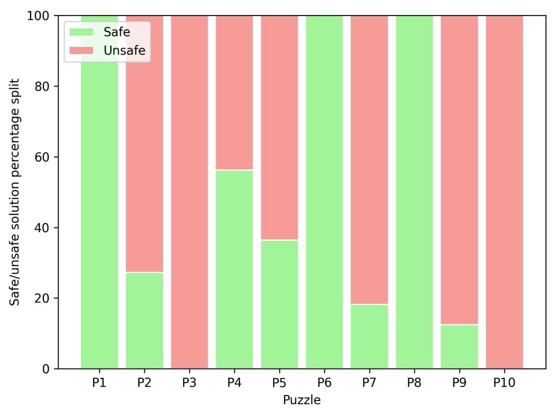


Figure 6.13: Rush Hour planning tasks used in Human-aware Intervention **without hints**

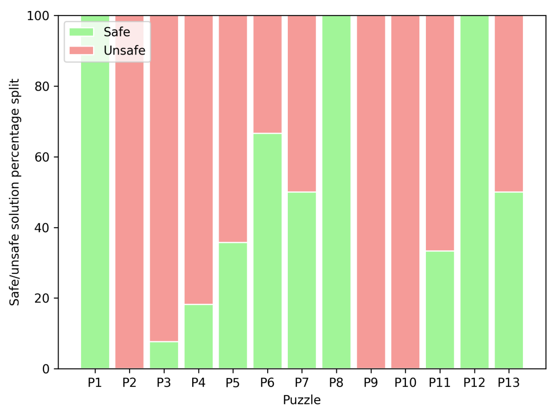


Figure 6.14: Rush Hour planning tasks used in Human-aware Intervention **with hints**

Table 6.7: Optimal costs for the Rush Hour planning tasks $P_{observer}$ and P_{user} optimized for number of moves and number of cars. $P_{observer}$ finds solutions without moving the forbidden vehicle. P_{user} may find a shorter solution while moving the forbidden vehicle.

PID	Type	Number of Moves		Number of Cars	
		$P_{observer}$	P_{user}	$P_{observer}$	P_{user}
P1	E	24	24	8	8
P2	C	30	30	10	10
P3	E	**35**	**25**	10	10
P4	C	23	23	9	9
P5	E	**21**	**14**	7	7
P6	C	22	22	8	8
P7	M	21	21	8	8
P8	C	9	9	5	5
P9	E	21	21	10	10
P10	E	24	24	9	9
P11	M	26	26	7	7
P12	M	17	17	8	8
P13	M	21	21	8	8

some cases, P_{user} may produce shorter plans than $P_{observer}$ as evidenced by cost optimal solutions for P3 and P5 when the number of moves is used as the optimizing criterion.

Figure 6.15 illustrates how the number of moves in human user solutions compare to threshold values derived from the optimal solution produced by a planner. The charts in the top row are human user solutions to type **E** puzzles, the middle row are type **C** puzzles and the bottom row are type **M** puzzles. We define the threshold α as the optimal number of moves obtained from the automated planner for $P_{observer}$. Then we derive a set of threshold values using α as the baseline such that $\{1.2 \times \alpha, 1.4 \times \alpha, 1.6 \times \alpha, 1.8 \times \alpha\}$. The higher threshold values indicate longer solutions.

In general, the human user solutions are longer than the cost optimal solution produced by a planner for the same planning task. However, for P8 puzzle, which had the forbidden car in the corner of the board, all but one user found cost optimal solutions. Human users have difficulty finding solutions closer to the optimal for P2. The same is observed when the human users solve P2 without hints. In our previous study, for puzzles P3 and P5 human users found solutions shorter than the optimal solution for $P_{observer}$. However, these solutions all require the user to move the forbidden vehicle. This is not the case when the users solve the planning task with hints. The reason for this behavior is that we modified the experiment conditions to block the user from moving the forbidden vehicle, which prevented the user from searching for the solution along the same path.

Using Human-aware Intervention with hints, 78% of the users who attempted the type E puzzles found solutions after the $1.4 \times \alpha$ threshold. For the type C puzzle, 60% of the users who attempted the puzzle found solutions after the $1.4 \times \alpha$ threshold. For the type M puzzle, this value is 45%. This finding indicate that human users take a long time to find solutions to the type E puzzles where the forbidden vehicle is on the edge of the board compared to the type C and type M. Human users find the solutions faster when the forbidden vehicle is in the middle of the board (type M).

When solving the Rush Hour planning task without hints, 56% of the users who attempted the type E found solutions after the $1.4 \times \alpha$ threshold. For the type C puzzle, 40% of the users who attempted the puzzle found solutions after the $1.4 \times \alpha$ threshold. For that experiment we only

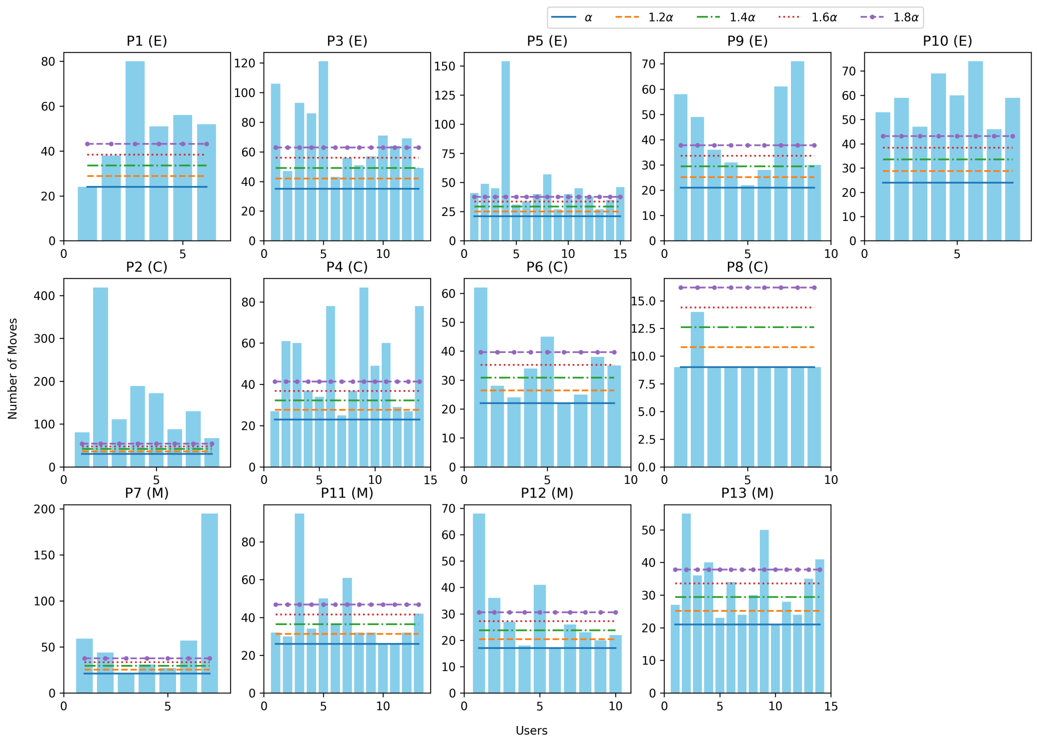


Figure 6.15: Number of moves in the human user solutions compared to the optimal number of moves α, and 1.2α, 1.4α, 1.6α, 1.8α number of moves for the puzzles P1 through P13. Within brackets next to the puzzle identifier is the puzzle type per Table 6.3

had one puzzle of the type M. In that puzzle 90% of the users found solutions after the $1.4 \times \alpha$ threshold. Similar to the case where they used hints, the time taken to find solutions to the type C puzzles is longer than the type M puzzles.

6.7.4 Relationship Between Avoiding the Forbidden Vehicle and the Number of Moves

When the human users solved the Rush Hour planning task without hints, we sorted the solutions by the number of moves in the ascending order and split them into three groups (fast, medium, slow). We ensured that the three groups for each puzzle contained approximately equal number of users. Table 6.8 summarizes the mean solution length in number of moves for each puzzle and the

Table 6.8: The mean number of moves and the frequency of forbidden moves in fast, medium and slow solution groups, when the Rush Hour planning task is **solved without hints**

PID	Fast		Medium		Slow	
	Mean (St. Dev.)	Forbidden Moves	Mean (St. Dev.)	Forbidden Moves	Mean (St. Dev.)	Forbidden Moves
P1	25.5 (1.5)	0	41.7 (7.0)	0	64.7 (21.1)	0
P2	74.7 (22)	4	137.7 (21)	16	277 (112.5)	28
P3	26.5 (1.9)	8	36.2 (4.3)	9	43.7 (5.6)	6
P4	25.2 (1.8)	1	32 (4.1)	4	76 (29.0)	28
P5	18.3 (4.0)	3	26 (1.0)	2	44.75 (24.9)	13
P6	22 (0)	0	24.5 (1.3)	0	39.6 (11.0)	0
P7	38.5 (7.2)	5	53.3 (8.5)	16	120 (11.7)	37
P8	9 (0)	0	9 (0)	0	10 (1.1)	0
P9	27.8 (3.8)	8	48.6 (10.0)	12	99 (38.7)	28
P10	50.3 (7.8)	11	66 (6.9)	14	127.3 (32.7)	46

number of times the forbidden vehicle was moved in fast, medium and slow groups. There were 46 users in the fast group, 42 in the medium and 48 in the slow group.

We can see that the longer the solutions are the more frequently the users will move the forbidden vehicle. Is this relationship statistically significant? In order to address this question, we first perform the normality test between the two variables: **number of moves** and the **number of times the forbidden car was moved**. We used the Shapiro-Wilks test for the two distributions with H_0 : the distribution is normally distributed, and H_A : the distribution is not normally distributed. Given $\alpha = 0.05$, Shapiro-Wilks test gives that the p-value < 0.05 for both forbidden vehicle moves and the number of moves. Thus we reject H_0 for both distributions.

To test the relationship between the number of moves and the forbidden vehicle moves when the hints are not used, we define the null and alternative hypotheses as follows. Since we showed that there is evidence that the distributions are not normally distributed, we use the non-parametric test Spearman's Rank Correlation to measure the strength of relationship. Let ρ_s be the Spearman's population correlation coefficient:

- $H_0 : \rho_s = 0$, there is no correlation between the **number of moves** and the **number of times the forbidden car was moved**

- $H_A : \rho_s \neq 0$, there is a correlation between the **number of moves** and the **number of times the forbidden car was moved**

Given $\alpha = 0.05$, Spearman's rank correlation coefficient is $\rho_s = 0.52$ and p-value < 0.05. Thus we reject the null hypothesis.

Next, we test whether the correlation between the solution length and the number of forbidden vehicle moves is the same when the human users solve the Rush Hour planning task with hints. We follow the same procedure as above to split the solutions into fast, medium and slow groups. Initially, there were 45, 42 and 48 solutions in each group respectively. However, there were instances where the observer agent did not display any hints during the planning task. We removed the subjects who did not see any hints for this analysis, leaving 20 users in the fast group, 26 users in the medium group and 35 users in the slow group. Table 6.9 shows the mean solution length and the count of forbidden vehicle moves for each puzzle.

To test the relationship between the number of moves and the forbidden vehicle moves when the hints are used, we define the null and alternative hypotheses the same as above. The test of normality using the Shapiro-Wilks test show that the two variables: number of moves and the number of times the forbidden vehicle was moved, are not normally distributed. Therefore, we use the Spearman's Rank Correlation test to measure the strength of the relationship between the two variables. Given $\alpha = 0.05$, Spearman's rank correlation coefficient is $\rho_s = 0.475$ and p-value < 0.05. Thus, we reject H_0.

We conclude that the longer the users spend searching for the solution to the Rush Hour planning task, the more likely they will move the forbidden car. This observation is the same for solving the Rush Hour planning problem with and without hints.

6.7.5 Usage of Hints in the Condition Group

In this section we analyze how the users requested hints and discuss the findings to address the question: **What are the most frequently used hints in different stages of the Rush Hour planning task?**

Table 6.9: The mean number of moves and the frequency of forbidden moves in fast, medium and slow solution groups when the Rush Hour planning task is **solved with hints**. A hyphen (-) indicates that the users who solved that puzzle did not see any hints

PID	Fast		Medium		Slow	
	Mean (St. Dev.)	Forbidden Moves	Mean (St. Dev.)	Forbidden Moves	Mean (St. Dev.)	Forbidden Moves
P1	31 (9.9)	0	51.5 (0.7)	0	68 (17)	0
P2	236 (10.7)	3	130	1	260 (138)	10
P3	47.5 (3.4)	4	61.5 (6.1)	5	95.4 (19.1)	6
P4	-	-	43.4 (10.9)	4	75.8 (11.3)	4
P5	31.7 (3.6)	1	40.8 (2.6)	5	76.5 (51.9)	2
P6	-	-	34	0	53.5 (12.0)	1
P7	29 (2.8)	0	50.5 (9.2)	3	127 (96.2)	2
P8	-	-	-	-	14	0
P9	22	1	-	-	66 (7.1)	5
P10	49 (3.8)	4	59 (0)	2	67.7 (7.1)	3
P11	-	-	-	-	68.7 (23.5)	5
P12	-	-	-	-	41	0
P13	25 (1.7)	0	31.7 (3.3)	2	44.4 (7.8)	17

From the 91 subjects who requested hints during the puzzle solving task 49 users (54%) requested less than 2 hints. 20 subjects requested between 3–5 hints (22%). There are 4 subjects each in 6–8 hint and 9–11 hint request categories. 14 subjects requested more than 11 hints (15%). In Table 6.10, we report the total number of requests for each hint made by users in the slow, medium and fast groups. "Undo" refers to the instances where the user actually moved the forbidden vehicle and saw the alert message in Figure 6.2. The "Undo" alert is seen by users in all three groups. In all three groups "Show the next best move" is the most requested hint. This finding corroborates the subjective helpfulness rating the users gave for the hints in the post-study survey where the hint "Show the next best move" was rated as the most helpful compared to the others. The second most requested option in all three groups is the "Ignore", where the users disregard the Human-aware intervention. The least used hint in the slow group is "Show the vehicles that must be moved". The same can be observed in the fast group. In the medium group, the least requested hint is "Restart". Intuitively, the users in the fast group did not request for the "Restart" hint. This is because we count the number of moves before and after the restart as the solution length and

Table 6.10: Number of times each hint was selected by users in the slow, medium and fast solution groups

Speed	Selected Hint					
	Undo	Remaining	Must Move	Next Best	Restart	Ignore
Slow	57	36	10	131	14	73
Medium	23	9	17	120	5	42
Fast	18	11	3	90	0	32

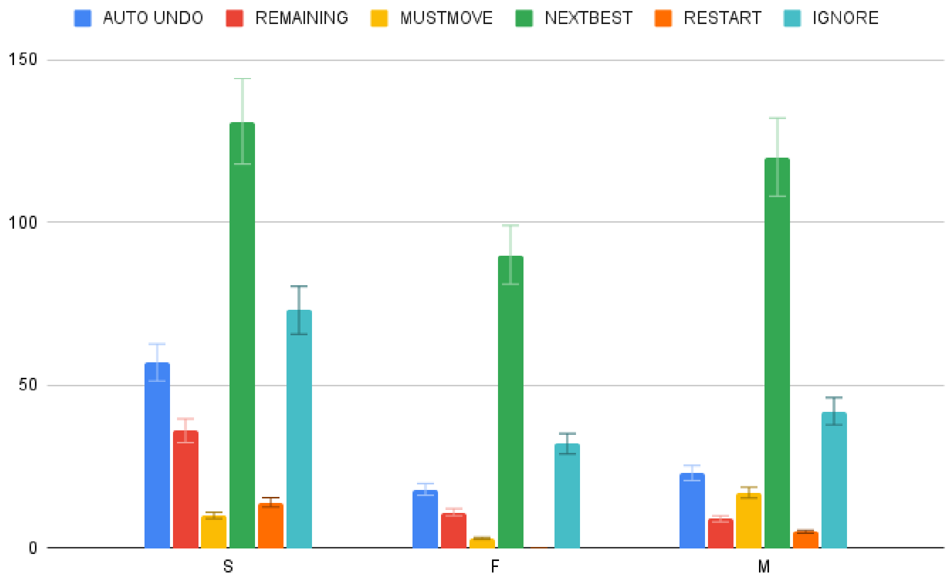


Figure 6.16: Number of requests for each hint type in slow (S), medium (M) and fast (F) solution groups.

restarting the puzzle might push the user out of the fast group. Figure 6.16 illustrates the request frequency distribution with the error bars.

We use the Fisher's exact test to verify if the hint selection is significantly related to the puzzle solving speed. The chi-squared test of independence can not be used in this situation because our contingency table (Table 6.10) contains a cell with a zero frequency. We define the null (H_0) and alternative (H_A) hypotheses as follows:

- H_0 : The two categorical variables, Selected Hint and Speed are independent.
- H_A : The two categorical variables, Selected Hint and Speed are dependent.

Given the significance level $\alpha = 0.05$, we reject H_0 ($p - value = 0.0005$) and conclude that there is a significant relationship between the hint selection and the speed.

Table 6.11: Number of times each hint was selected by users in the Rush Hour planning task types E, M, C

Type	Selected Hint					
	Undo	Remaining	Must Move	Next Best	Restart	Ignore
E	40	18	6	145	12	66
M	31	13	6	100	2	38
C	27	25	18	96	5	43

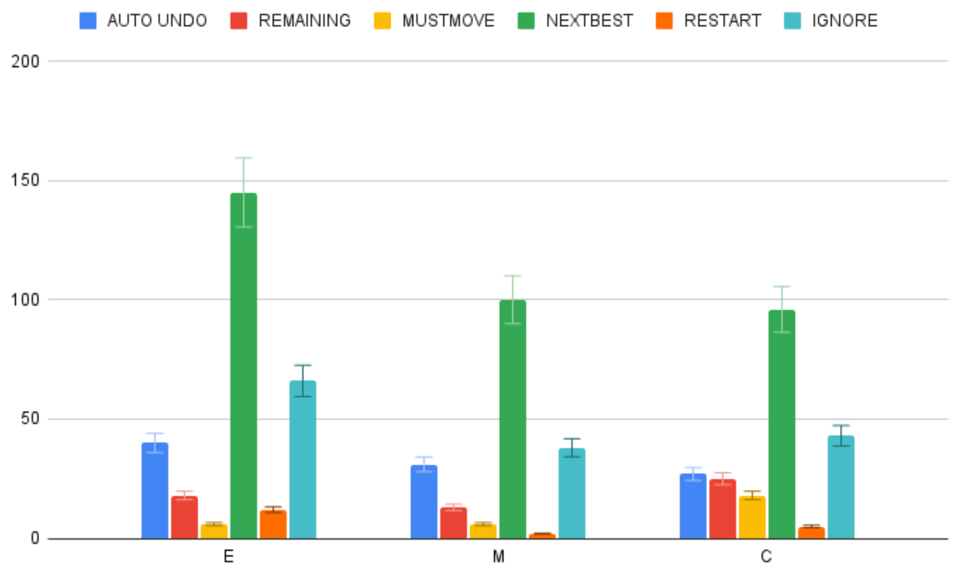


Figure 6.17: Number of requests for each hint in the Rush Hour planning task types E, M, C with error bars.

We now analyze the hint request patterns considering the three Rush Hour planning task types E, M, C. In Table 6.11, we report the sum of requests for the hints considering the three planning task types. For each planning task types, the most requested hint is “Show the next best move”. The same was observed when considering user’s solution speed. The second most requested option is “Ignore”. The least requested hint is “Restart” for type C and M planning tasks. For type E, the least requested hint is “Show the vehicles that must be moved”. The “Undo” option was triggered for all three planning tasks. Figure 6.17 illustrates the request frequency distribution with the error bars.

We perform the chi-square test for independence to verify if the hint selection is significantly related to the type of puzzle. We define the null (H_0) and alternative (H_A) hypotheses as follows:

- H_0 : The two categorical variables, Selected Hint and Planning Task Type are independent.

Table 6.12: The total number of hints requested from each type sorted by the request number - part 1

Request Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Undo	57	10	6	6	3	4	2	2	2	1	1	1	0	1	0	0	0	0	0	0
Remaining	6	5	10	6	3	4	1	0	1	1	1	3	2	1	1	1	1	1	0	1
Next Best	14	35	27	22	20	18	15	17	16	17	14	12	12	10	9	8	8	8	8	7
Must Move	4	5	4	3	3	2	1	1	0	0	2	0	1	1	1	1	1	0	0	0
Ignore	8	19	23	19	13	10	11	5	4	2	2	3	2	3	3	3	1	1	2	1
Restart	2	6	2	4	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Total	91	80	72	60	47	38	30	25	23	21	20	19	17	16	14	13	11	10	10	9

Table 6.13: The total number of hints requested from each type sorted by the request number - part 2

Request Number	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
Undo	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Remaining	0	1	0	0	1	0	1	1	1	0	1	1	0	1	1	1	0	0
Next Best	6	6	6	4	4	4	4	3	3	1	1	1	1	0	0	0	0	0
Must Move	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ignore	2	0	1	3	1	1	0	0	0	2	1	0	1	0	0	0	1	1
Restart	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Total	8	7	7	7	6	5	5	4	4	3	3	2	2	1	1	1	1	1

- H_A : The two categorical variables, Selected Hint and Planning Task type are dependent.

Given the significance level $\alpha = 0.05$, we reject H_0 ($p - value = 0.007, \chi^2 = 24.4, df = 10$) and conclude that there is a significantly relationship between the hint selection and the planning task type.

Figure 6.18 illustrates the percentage split between the hints the subjects selected each time the Human-aware Intervention agent recognized that the user needed help. The x-axis, Request Number refers to the sequence of requests in the ascending order. For example, Request Number 1 means the first choice the subjects made when the Human-aware Intervention agent recognized that the user needed intervention, Request Number 2 is the second choice the subjects made and so on. The maximum number of hints requested by a user is 38. Tables 6.12 and 6.13 show the counts of each hint requested. The first hint seen by majority of the users (63%) is the alert message indicating the forbidden vehicle is moved. One reason for this observation is that we set a threshold for the number of steps for the Human-aware Intervention agent to start displaying the hints. For each planning task, we use lower value of the number of moves in the cost optimal solution to P_{user} or $P_{observer}$ as the threshold (see Table 6.7). For all planning tasks except P3 and P5 the

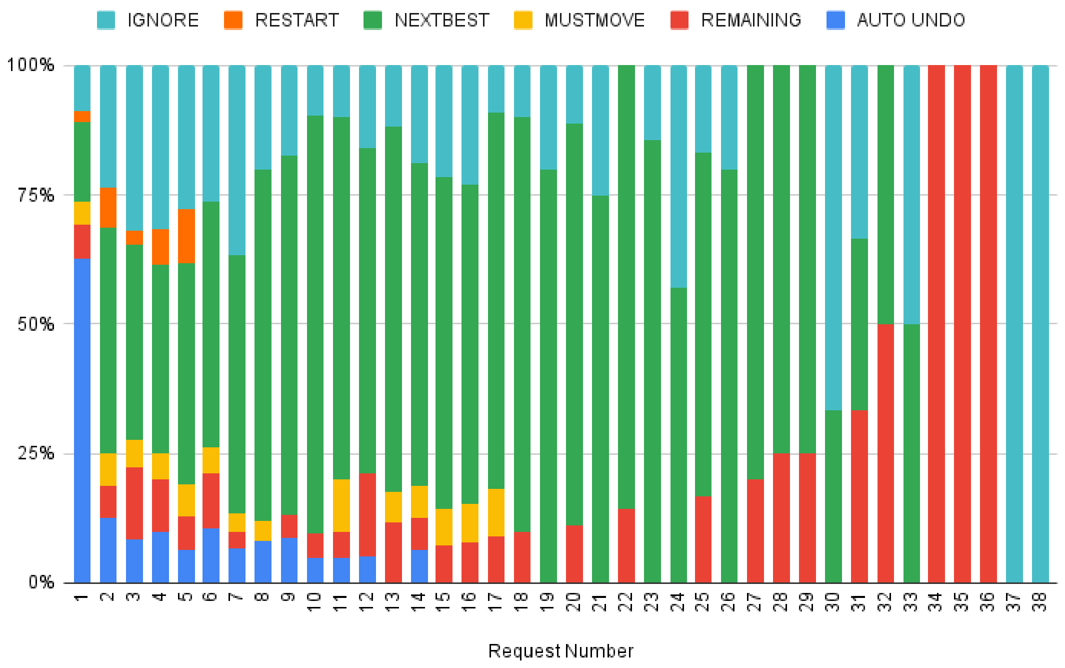


Figure 6.18: Percentage split for hints for each request number

number of moves in the cost optimal solutions to P_{user} or $P_{observer}$ are the same. In P3 and P5 the solution to P_{user} is shorter than the solution to $P_{observer}$. The threshold is required to ensure that the logistic regression classifier we use to recognize intervention has enough data to be used as input in the beginning. This means that the Human-aware Intervention agent does not perform any recognition from the start of the planning task until the threshold is exceeded. From the 57 subjects who saw the forbidden vehicle move alert message as the first hint, only 5 subjects (1 user who solved P2, 2 users who solved P4, 1 user who solved P5 and 1 user who solved P7) saw it after they had gone past the corresponding thresholds. The remaining 52 subjects missed the interaction with the Human-aware Intervention agent due to the constraint in the machine learning method we use in this study. The other reason could be that the subjects had difficulty recognizing the forbidden vehicle on their own although they were primed to think about its presence. The Table F.1 in Appendix F summarizes the raw data for the solutions produced by the 57 subjects.

Figure 6.18 further corroborates the finding that the hint “Show next best move” is the most requested hint. Moreover, it is selected by the subjects as the early hint choices as well as choices at the end. The hint “Show the remaining number of moves” is requested less often at first and more often towards the end. The hint “Show the vehicles that must be moved” is only requested early. The hint “Ignore” is chosen during all stages of the planning task. The reason for this observation is because we leave it to the subject’s discretion to decide when they want to turn off the Interactive Human-aware Intervention agent. The hint “Restart” is used only in early requests. It is also the least requested hint.

To examine how the hints are requested for each planning task, we consider the first ten hint requests and the mean remaining number of steps in the planning task at each request. The mean remaining number of steps for a hint $\mathcal{N}$ for the i^{th} request $i = \{1, \ldots, 10\}$ during a planning task is computed as follows:

$$\frac{\sum(\text{number of steps in solution}) - (\text{number of steps before request } \mathcal{N})}{\text{number of users requesting } \mathcal{N}} \tag{6.1}$$

The mean remaining number of steps indicates how early in the planning task a particular hint is requested. We limit the analysis to the first ten hint requests because that threshold covers the majority of the subjects who used hints. Recall that only 14 subjects requested more than 11 hints. We breakdown the findings by planning task type: C, E and M.

Figure 6.19 illustrates how users request hints in type C planning tasks. Table F.2 in the Appendix F contains the data used to generate the graphs in Figure 6.19. We have seen in Figure 6.15, that the users who solved the planning task P2 took a long time to find the solution. In contrast, users who solved the P8 planning task found solutions quicker and closer to the number of moves in the cost optimal solution. The subjects attempting the P2 planning task have moved the forbidden vehicle multiple times (“Undo” hint in many requests). They have also requested the “Show the next best move” hint in all ten requests because they were having difficulty finding solution for the puzzle. The “Ignore” option is chosen only in the first six requests. The “Restart” option is chosen only once. The hint “Show the remaining number of moves” is chosen only once, very early in the planning task (mean remaining moves 387). The maximum number of moves to solve P2 is 419. In contrast, the subjects attempting the P8 planning task only requested hints 4 times. In

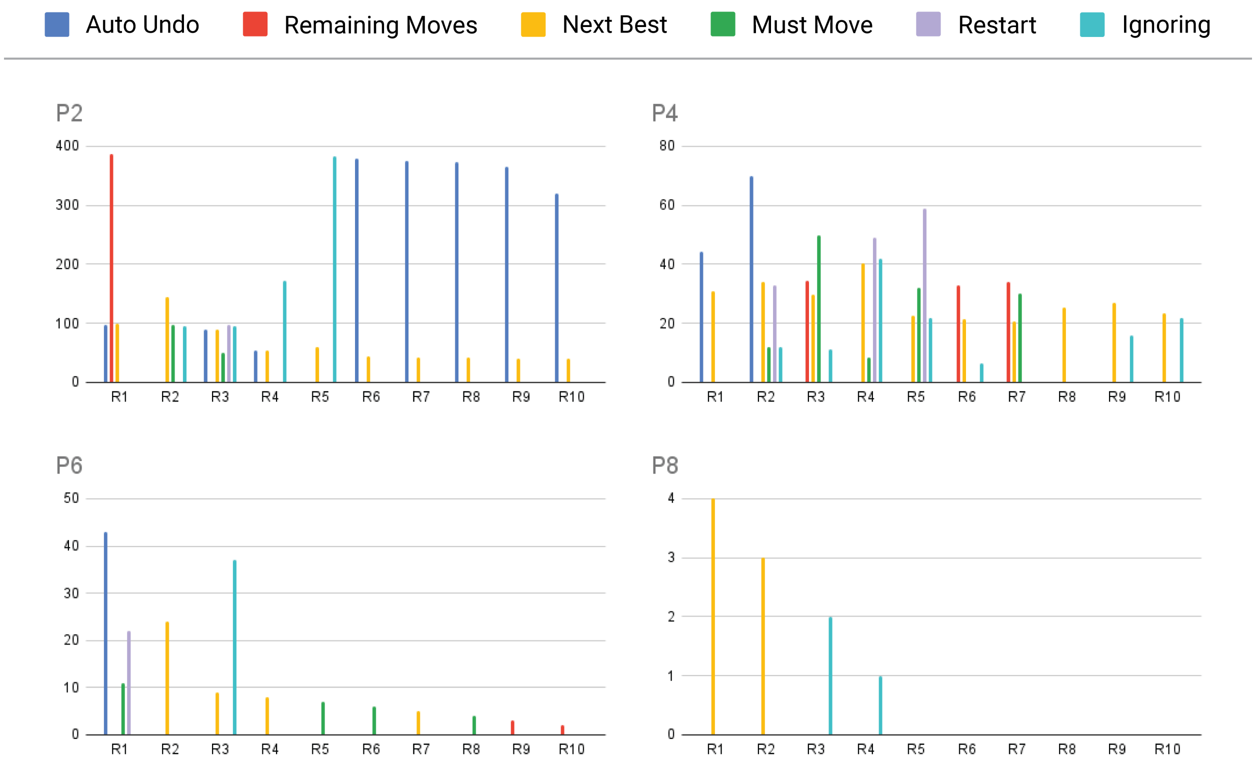


Figure 6.19: Hint request distribution for type C planning tasks

this case they only requested the "Show the next best move" hint early on with 4 and 3 remaining number of steps on average. Later they turned off the hints when they were 2 and 1 move away from the solution on average. The subjects who attempted the P4 and P6 planning tasks saw the "Undo" hint early. They also requested the "Show the next best move" hint throughout the puzzle solving task. The subjects attempting the P4 and P6 planning tasks used the hint "Show the vehicles that must be moved" more often compared to the subjects who attempted P2 and P8. The subjects attempting the P4 planning task used the hints "Show the remaining number of moves" more often than subjects attempting P2, P6 and P8. For the planning task P4, the subjects requested restarts when there the mean number of moves remaining is between 50 and 60.

Figure 6.20 illustrates how users request hints in type C planning tasks. Table F.3 in the Appendix F contains the data used to generate the graphs in Figure 6.20. Similar to the hint distribution in type C planning tasks, subjects attempting type E tasks also request the hint "Show the next best move" in the first ten requests. The hint "Show the remaining number of moves" is requested

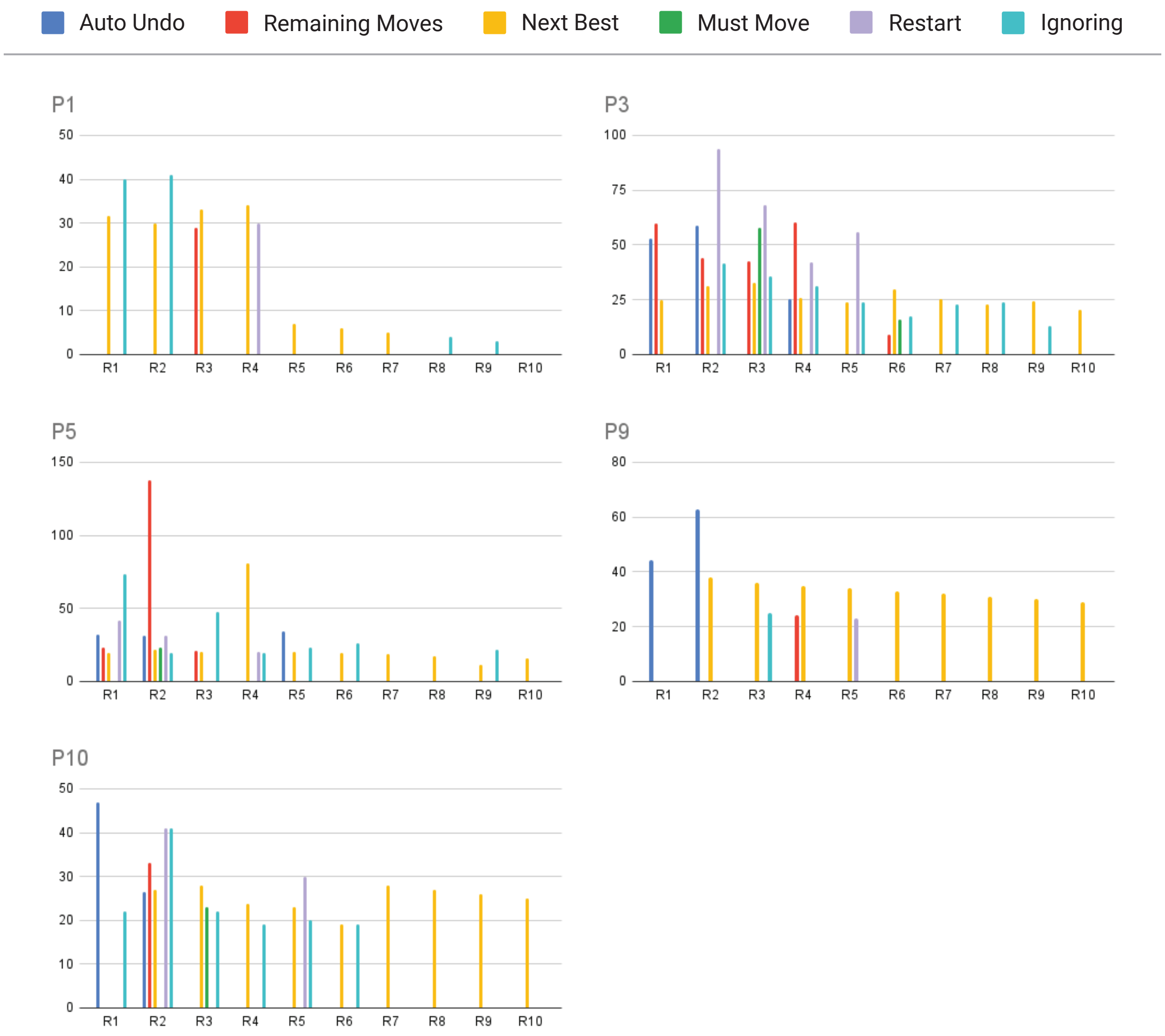


Figure 6.20: Hint request distribution for type E planning tasks

between the first and the fifth request. However, for planning tasks P3 and P5, this hint request appears more often. The hint "Restart" is also requested between the first and the fifth request. For the P3 and P5 planning tasks the "Ignore" hint is requested throughout the first ten requests. The subjects attempting type E planning tasks also moved the forbidden vehicle multiple times early on.

Figure 6.21 illustrates how users request hints in type M planning tasks. Table F.4 in the Appendix F contains the data used to generate the graphs in Figure 6.21. Similar to type C and E,

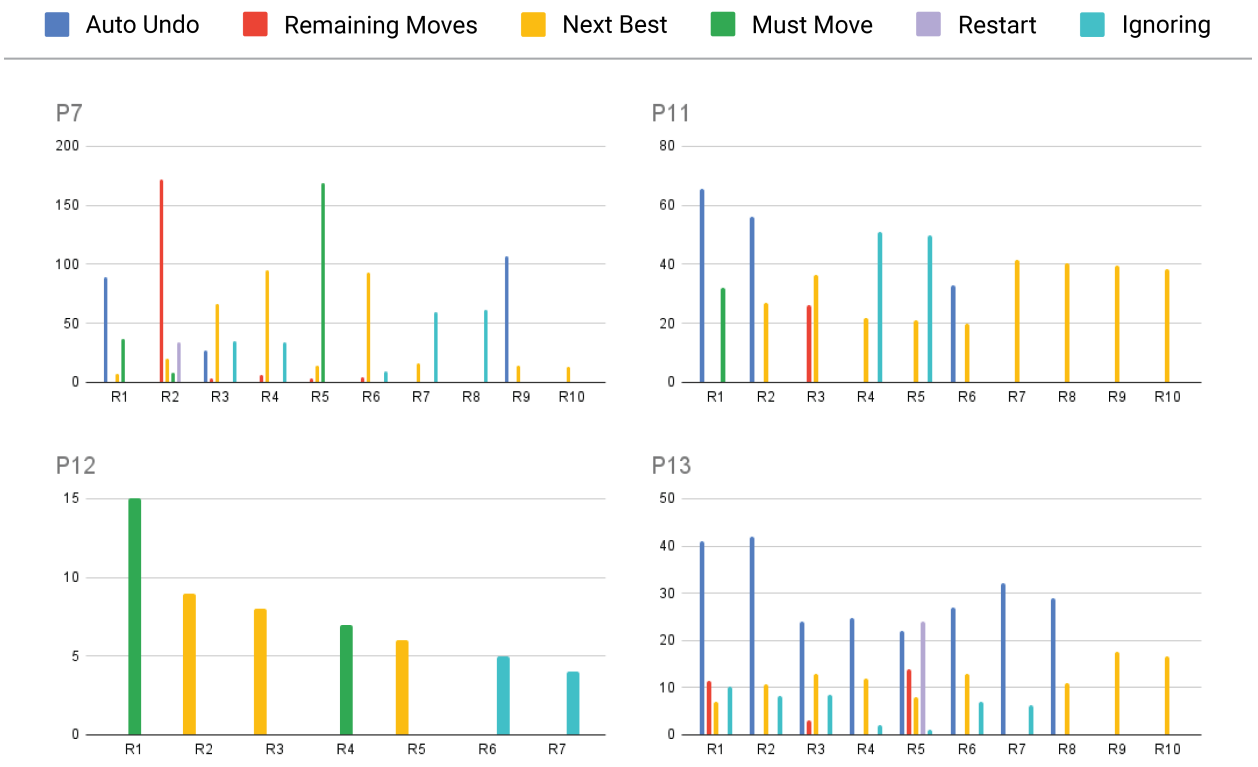


Figure 6.21: Hint request distribution for type M planning tasks

the hint “Show the next best move” is requested often in type M planning tasks. For the planning task P7, the hint “Show the remaining number of moves is requested more often compared to the other planning tasks in type M. The “Restart” hint is not a common choice among subjects who attempted type M puzzles. Only users attempting P13 planning task requested a restart. The users attempting P13 planning task also moved the forbidden vehicle multiple times. For the planning task P12, we only have one subject using the hints. The subject requested the hint “Show the vehicles that must be moved” early in the planning task. Then the user requested the “Show the next best move” hint 3 times and closer toward the end of the task opted to turn off the hints.

6.7.6 The Effectiveness of the Interactive Human-aware Intervention

In Section 6.7.4, we conclude that in order help the user avoid the forbidden vehicle, the Interactive Human-aware Intervention agent must direct the user toward the cost optimal solution for the planning task. In this section, we evaluate how successful the interactive Human-aware Intervention model has been in moving the user's solution closer to the cost optimal solution using the hints. The hints are properties derived from the cost optimal solution and the fact landmarks of $P_{observer}$. Recall that the fact landmarks are state predicates that must be true in any valid solution plan for $P_{observer}$ (Hoffmann et al., 2004). We want to address the following:

Evaluation Questions

1. Does the Interactive Human-aware Intervention have an effect on the solution length?
2. Does the Interactive Human-aware Intervention help move the user closer to the optimal solution?
3. Does seeing a help video affect the solution length?

We introduce the following metrics to resolve the evaluation questions.

Evaluation Metrics

1. The number of moves in the human user's solution
2. The difference from the cost optimal solution
3. The latest time a fact landmark is eventually achieved (landmark achievement)
4. The number of times a fact landmark is lost and regained (landmark regain)

The definitions for these metrics will be presented later. Question 1 is answered using the evaluation metric 1. Question 2 is answered using the evaluation metrics 2, 3 and 4. Question 3 is answered using the evaluation metric 1.

For questions 1 and 2, we use the data from the control and condition groups for the planning tasks P1, P2, P4, P6, P7, P8, P9 and P10 for the evaluation. We remove the planning tasks P3 and P5 because for those two tasks, the cost optimal solution while moving the forbidden vehicle is shorter than the cost optimal solution without moving the forbidden vehicle. We also remove the planning tasks P11, P12 and P13 because these tasks are only used in the condition group. For question 3 we use the data from the condition group and include the planning tasks P1 through P13.

Using the Landmarks

To derive the landmark based evaluation metrics, we examine how landmarks are achieved in cost optimal solutions generated by an automated planner for a given Rush Hour planning task. We use the algorithm advanced by Hoffmann et al. (2004) to generate the *Greedy Necessary Fact Landmarks* for the planning task $P_{observer}$ and the actions that have the landmarks as post-conditions. The algorithm uses a data structure called the Landmark Generation Graph (LGG) to find the landmarks and the greedy necessary orders between them. Nodes in LGG are the fact landmarks and the edges are the greedy necessary orders. Starting from the goal (the first landmark candidates), the algorithm uses the back-chaining process to find the earliest actions that can be used to achieve each landmark. Here, early means a greedy approximation of reachability from the initial state. The landmark achievers are actions grounded with the objects that must be moved to solve the planning task from the current state.

In order to analyze how the landmarks are achieved in the solution, we use the Fast Downward planner configured with the admissible heuristic Landmark-cut (`lmcut`) to generate 100 cost optimal plans for each planning task. Figure 6.22 illustrates how landmarks are achieved in five cost optimal plans for the planning tasks P1, P2, P4, P6, P7, P8, P9 and P10. We manually examine the 100 cost optimal plans for each planning task identify frequently occurring landmark achievement patterns. We see that in all cost optimal solutions the landmarks are achieved in *clusters*. We use the notation $[Lj, \ldots]$ to represent a **landmark cluster**, where $Lj, j \in \mathbb{N}$ refers to the landmark identifier. Actions that are not landmarks that appear in the cost optimal plan are denoted with the identifier $(Ak, \ldots), k \in \mathbb{N}$. Actions that the users execute, but do not appear in the cost optimal plans are denoted with the identifier $(Ul, \ldots), l \in \mathbb{N}$. The landmark and action identifiers are unique to the planning task. We denote this as $\langle Pi \rangle [Lj, \ldots](Ak, \ldots)$, where Pi refers to the planning task identifier, $i = \{1, 2, 4, 6, 7, 8, 9, 10\}$.

There are groups of actions that achieve some landmarks early in the solution. For example, consider the cost optimal solutions for the planning task P1 in Figure 6.22. The cost optimal solution for P1 has 24 moves. The cluster $[L1, L2, L3, L4, L5, L6]$ in the planning task P1 appear

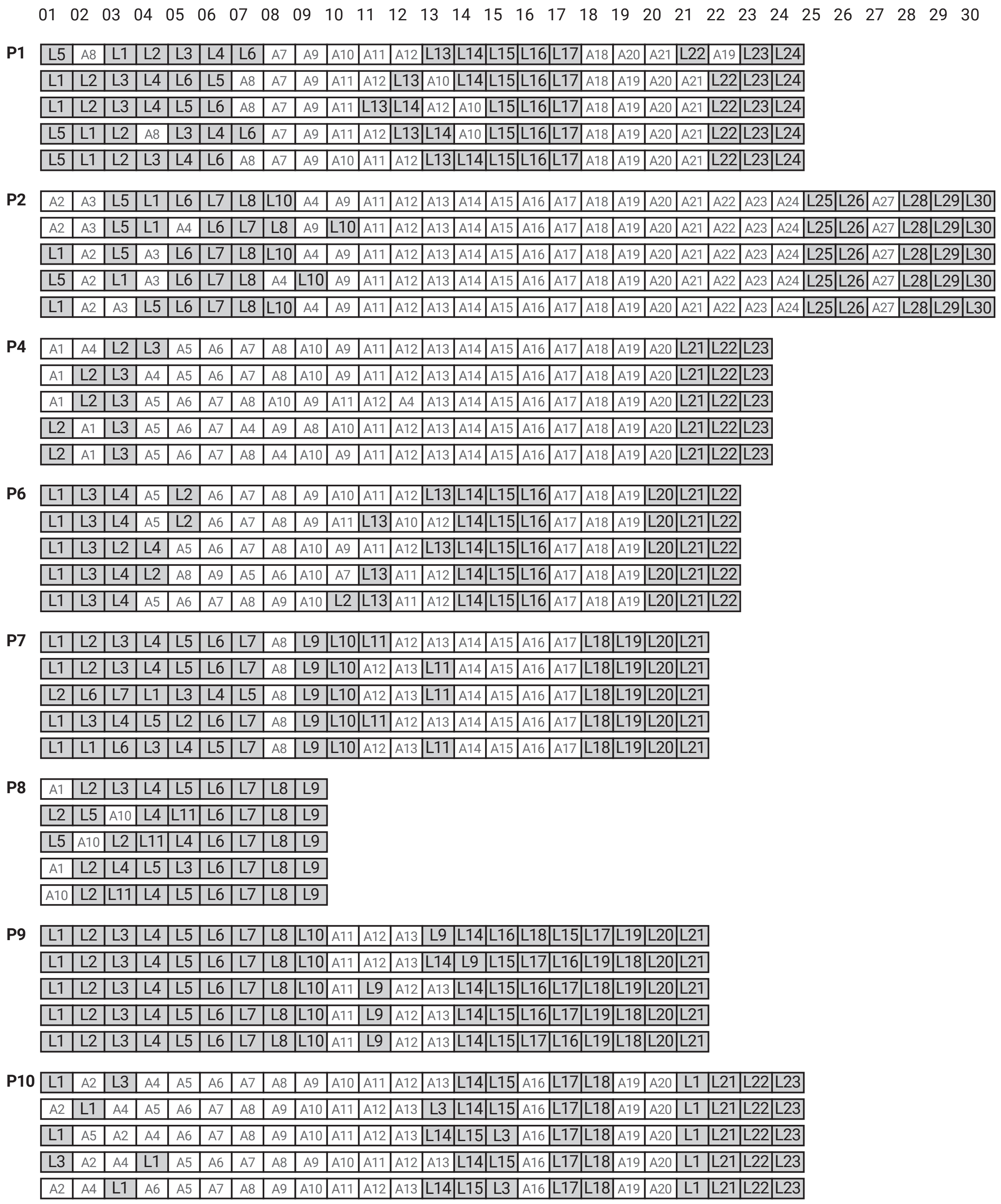

01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
P1
L5 A8 L1 L2 L3 L4 L6 A7 A9 A10 A11 A12 L13 L14 L15 L16 L17 A18 A20 A21 L22 A19 L23 L24
L1 L2 L3 L4 L6 L5 A8 A7 A9 A11 A12 L13 A10 L14 L15 L16 L17 A18 A19 A20 A21 L22 L23 L24
L1 L2 L3 L4 L5 L6 A8 A7 A9 A11 L13 L14 A12 A10 L15 L16 L17 A18 A19 A20 A21 L22 L23 L24
L5 L1 L2 A8 L3 L4 L6 A7 A9 A11 A12 L13 L14 A10 L15 L16 L17 A18 A19 A20 A21 L22 L23 L24
L5 L1 L2 L3 L4 L6 A8 A7 A9 A10 A11 A12 L13 L14 L15 L16 L17 A18 A19 A20 A21 L22 L23 L24
P2
A2 A3 L5 L1 L6 L7 L8 L10 A4 A9 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 A21 A22 A23 A24 L25 L26 A27 L28 L29 L30
A2 A3 L5 L1 A4 L6 L7 L8 A9 L10 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 A21 A22 A23 A24 L25 L26 A27 L28 L29 L30
L1 A2 L5 A3 L6 L7 L8 L10 A4 A9 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 A21 A22 A23 A24 L25 L26 A27 L28 L29 L30
L5 A2 L1 A3 L6 L7 L8 A4 L10 A9 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 A21 A22 A23 A24 L25 L26 A27 L28 L29 L30
L1 A2 A3 L5 L6 L7 L8 L10 A4 A9 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 A21 A22 A23 A24 L25 L26 A27 L28 L29 L30
P4
A1 A4 L2 L3 A5 A6 A7 A8 A10 A9 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 L21 L22 L23
A1 L2 L3 A4 A5 A6 A7 A8 A10 A9 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 L21 L22 L23
A1 L2 L3 A5 A6 A7 A8 A10 A9 A11 A12 A4 A13 A14 A15 A16 A17 A18 A19 A20 L21 L22 L23
L2 A1 L3 A5 A6 A7 A4 A9 A8 A10 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 L21 L22 L23
L2 A1 L3 A5 A6 A7 A8 A4 A10 A9 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 L21 L22 L23
P6
L1 L3 L4 A5 L2 A6 A7 A8 A9 A10 A11 A12 L13 L14 L15 L16 A17 A18 A19 L20 L21 L22
L1 L3 L4 A5 L2 A6 A7 A8 A9 A11 L13 A10 A12 L14 L15 L16 A17 A18 A19 L20 L21 L22
L1 L3 L2 L4 A5 A6 A7 A8 A10 A9 A11 A12 L13 L14 L15 L16 A17 A18 A19 L20 L21 L22
L1 L3 L4 L2 A8 A9 A5 A6 A10 A7 L13 A11 A12 L14 L15 L16 A17 A18 A19 L20 L21 L22
L1 L3 L4 A5 A6 A7 A8 A9 A10 L2 L13 A11 A12 L14 L15 L16 A17 A18 A19 L20 L21 L22
P7
L1 L2 L3 L4 L5 L6 L7 A8 L9 L10 L11 A12 A13 A14 A15 A16 A17 L18 L19 L20 L21
L1 L2 L3 L4 L5 L6 L7 A8 L9 L10 A12 A13 L11 A14 A15 A16 A17 L18 L19 L20 L21
L2 L6 L7 L1 L3 L4 L5 A8 L9 L10 A12 A13 L11 A14 A15 A16 A17 L18 L19 L20 L21
L1 L3 L4 L5 L2 L6 L7 A8 L9 L10 L11 A12 A13 A14 A15 A16 A17 L18 L19 L20 L21
L1 L1 L6 L3 L4 L5 L7 A8 L9 L10 A12 A13 L11 A14 A15 A16 A17 L18 L19 L20 L21
P8
A1 L2 L3 L4 L5 L6 L7 L8 L9
L2 L5 A10 L4 L11 L6 L7 L8 L9
L5 A10 L2 L11 L4 L6 L7 L8 L9
A1 L2 L4 L5 L3 L6 L7 L8 L9
A10 L2 L11 L4 L5 L6 L7 L8 L9
P9
L1 L2 L3 L4 L5 L6 L7 L8 L10 A11 A12 A13 L9 L14 L16 L18 L15 L17 L19 L20 L21
L1 L2 L3 L4 L5 L6 L7 L8 L10 A11 A12 A13 L14 L9 L15 L17 L16 L19 L18 L20 L21
L1 L2 L3 L4 L5 L6 L7 L8 L10 A11 L9 A12 A13 L14 L15 L16 L17 L18 L19 L20 L21
L1 L2 L3 L4 L5 L6 L7 L8 L10 A11 L9 A12 A13 L14 L15 L16 L17 L19 L18 L20 L21
L1 L2 L3 L4 L5 L6 L7 L8 L10 A11 L9 A12 A13 L14 L15 L17 L16 L19 L18 L20 L21
P10
L1 A2 L3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 L14 L15 A16 L17 L18 A19 A20 L1 L21 L22 L23
A2 L1 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 L3 L14 L15 A16 L17 L18 A19 A20 L1 L21 L22 L23
L1 A5 A2 A4 A6 A7 A8 A9 A10 A11 A12 A13 L14 L15 L3 A16 L17 L18 A19 A20 L1 L21 L22 L23
L3 A2 A4 L1 A5 A6 A7 A8 A9 A10 A11 A12 A13 L14 L15 A16 L17 L18 A19 A20 L1 L21 L22 L23
A2 A4 L1 A6 A5 A7 A8 A9 A10 A11 A12 A13 L14 L15 L3 A16 L17 L18 A19 A20 L1 L21 L22 L23

Figure 6.22: Landmark achieving patterns in five cost optimal solutions generated by the automated planner. The figures are scaled to reflect the solution length. Actions that achieve the landmarks are in gray cells. Other actions are in white cells. Note that the landmark identifiers L1, L2, . . . are unique to the planning task.

very early in the cost optimal solution. The cluster $[L22, L23, L24]$ appear at the end most of the time. The cluster $[L15, L16, L17]$ appear together in the middle after $L14$ has been achieved. The cluster $[L13, L14]$ appear together in the middle before the cluster $[L15, L16, L17]$. Note that within a cluster there may be permutations of the order in which the individual landmarks are reached. For example, in the solutions for P1, the first cluster may be $[L1, L2, L3, L4, L5, L6]$ or $[L5, L1, L2, L3, L4, L6]$. To account for the permutations of landmarks in the solutions for the same planning task, we define a condensed representation considering the clustering patterns in 75 of the 100 solutions for a given planning task. The condensed representation applied to the cost optimal plans for the planning tasks P1, P2, P4, P6, P7, P8, P9, P10 are as follows:

- $\langle P1\rangle[L1, L2, L3, L4, L5, L6](A8)(A7, A9)(A12)[L13, L14](A10)$
  $[L15, L16, L17](A18, A19, A20, A21)[L22, L23, L24]$
- $\langle P2\rangle[L1, L5, L6, L7, L8, L10](A2, A3, A4, A9)$
  $(A11, A12, A13, A14, A15, A16, A17, A18, A19, A20, A21, A22, A23, A24)$
  $[L25L26](A27)[L28, L29, L30]$
- $\langle P4\rangle[L2, L3]$
  $(A1, A4, A5, A6, A7, A8, A9, A10, A11, A12, A13, A14, A15, A16, A17, A18, A19, A20)$
  $[L21, L22, L23]$
- $\langle P6\rangle[L1, L2, L3, L4](A5, A6, A7, A8, A9, A10)$
  $(A11, A12)[L13][L14, L15, L16](A17, A18, A19)[L20, L21, L22]$
- $\langle P7\rangle[L1, L2, L3, L4, L5, L6, L7, L9, L10](A8)[L11]$
  $(A12, A13, A14, A15, A16, A17)[L18, L19, L20, L21]$
- $\langle P8\rangle(A1, A10)[L2, L3, L4, L5, L6, L7, L8, L9, L10]$
- $\langle P9\rangle[L1, L2, L3, L4, L5, L6, L7, L8, L9, L10](A11, A12, A13)$
  $[L14, L15, L16, L17, L18, L19, L20, L21]$

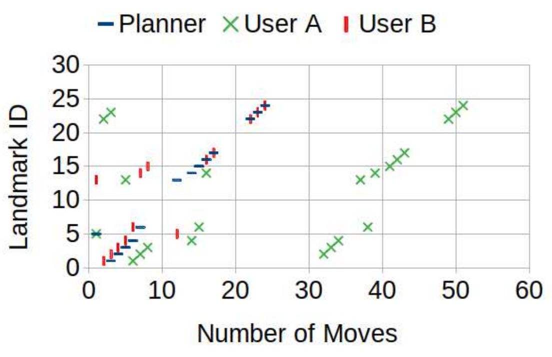


Figure 6.23: Human users achieving landmarks for the planning task P1

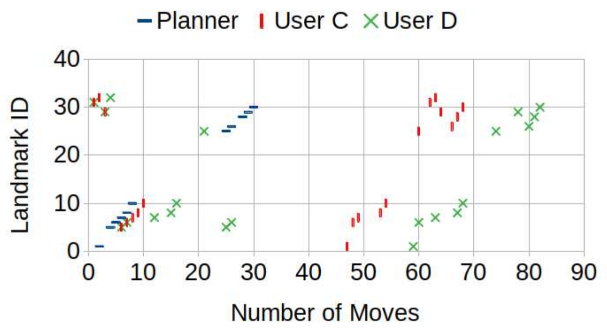


Figure 6.24: Human users achieving landmarks for the planning task P2

- $\langle P10\rangle[L1, L3](A2, A4, A5)(A6, A7, A8, A9, A10, A11, A12, A13)[L13, L14, L15](A16)$ $[L17, L18](A19, A20)[L1, L21, L22, L23]$

Although we can extract common patterns in achieving landmarks when the planning task is solved by an automated planner, this is not the case when human users solve planning tasks. Figure 6.23 illustrates how two human subjects: A and B achieve the landmarks for the planning task P1 compared to an automated planner. User B has achieved the landmarks for P1 similar to the automated planner. There are also a few instances where the same landmark was lost and regained later (i.e. same y-value for different x-values). Because user B achieved the landmarks similar to the automated planner, the user has been able to solve the planning task in fewer number of moves compared to the user A. In contrast, user A has taken more number of moves to achieve the landmarks. Consequently, user A's solution is longer. Both user A and B has eventually achieved the landmark clusters similar to the automated planner. Figure 6.24 illustrates how human subjects C and D achieve the landmarks for the planning task P2. Here too, we can see how the landmark clusters are distributed within the solution produced by the automated planner. We can see that user C has achieved the early landmark cluster similar to the automated planner. However, some actions the user executed have caused him/her to lose those landmarks and having to regain them later around the 50^{th} move. User D has lost the early landmark cluster twice before eventually regaining them around the 60^{th} move. User C's solution is shorter compared to user D. Therefore, when evaluating the human user solutions we monitor only the planning task's landmarks for when they are achieved for the last time and also how often they are regained.

We modify the condensed plan representation discussed above to include the two landmark evaluation metrics: (1) the latest time a fact landmark is eventually achieved and (2) the number of times a fact landmark is lost and regained. We remove the actions that are not landmarks that appear in the cost optimal plan, i.e., $(Ak, \ldots), k \in \mathbb{N}$ as well as the actions that the users execute, but do not appear in the cost optimal plans, i.e., $(Ul, \ldots), l \in \mathbb{N}$. The landmark cluster notation is modified as $[Lj(n), \ldots]$, where $Lj, j \in \mathbb{N}$ refers to the landmark identifier and $n \in \mathbb{N}$ corresponds to the evaluation metric value. For example, if we use the new condensed plan representation for user A's solution to denote the latest time the landmarks are achieved, it will be as follows:

$$\langle P1\rangle[L1(6), L2(32), L3(33), L4(34), L5(1), L6(38)][L13(37), L14(39)] \\ [L15(41), L16(42), L17(43)][L22(49), L23(50), L24(51)] \quad (6.2)$$

Using the new condensed plan representation for user A's solution to denote the number of times the landmarks are lost and regained:

$$\langle P1\rangle[L1(1), L2(2), L3(2), L4(2), L5(1), L6(2)][L13(2), L14(2)] \\ [L15(1), L16(1), L17(1)][L22(2), L23(2), L24(1)] \quad (6.3)$$

The latest times when the landmarks are achieved by the human subjects attempting the planning tasks P1, P2, P4, P6, P7, P8, P9 and P10 with the Interactive Human-aware Intervention are shown in Tables F.5, F.7, F.9, F.11, F.13, F.15, F.17, and F.19 respectively. The number of times a landmark is lost and regained for the planning tasks are shown in Tables F.6, F.8, F.10, F.12, F.14, F.16, F.18, and F.20.

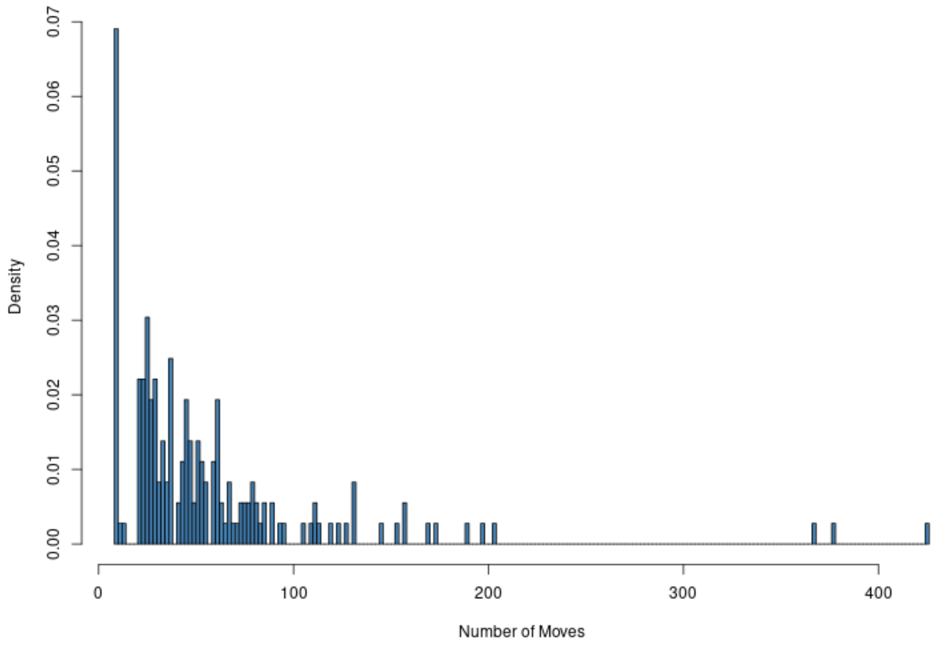


Figure 6.25: Frequency probability densities for the number of moves distribution

Data Preparation for Evaluation Question 1

We use the R statistical software for our analysis. The control group has 113 observations. The condition group has 68 observations. Figure 6.25 illustrates the frequency probability distribution of the number of moves in the solutions produced by the human users in the control and the condition groups. The data is highly skewed with a skewness factor +3.4. Because the sample size 181 $(113 + 68)$ is greater than the recommended size by the Central Limit Theorem, we assume that this is a normal distribution.

Effect on the Number of Moves

We address the question "*Does using the Interactive Human-aware Intervention (i.e., the hints) have an effect on the number of moves between the control and the condition groups?*" We propose the following study design.

- **Dependent variable**: Number of moves
- **Independent variables:**

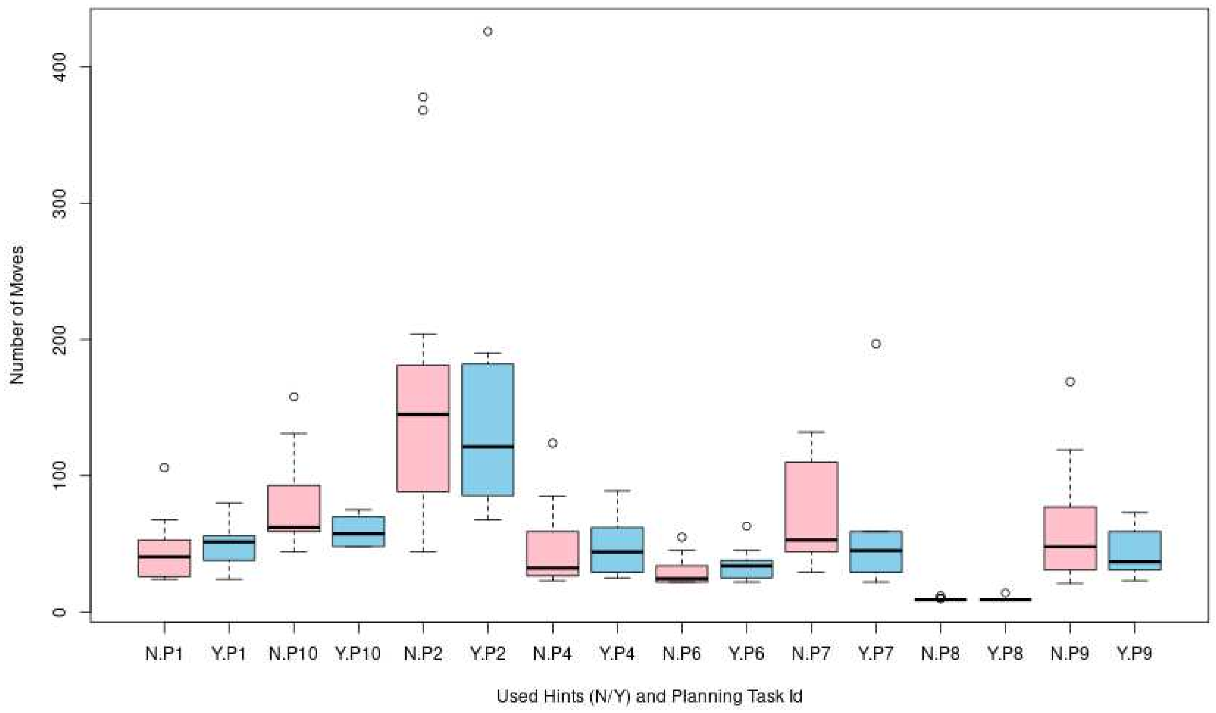


Figure 6.26: Box plots for the number of moves in the condition (Y) and the control (N) group for each planning task

- Used the Interactive Human-aware Intervention (Yes / No)
- Planning Task Id (P1, P2, P4, P6, P7, P8, P9, P10)
- Planning Task Type (C, E, M)

In Figure 6.26, the blue box plots represent the human subjects who used the Interactive Human-aware Intervention, while the red box plots represent the human subjects who did not. We see that human subjects who used the Interactive Human-aware Intervention have a higher median number of moves for planning tasks such as P1, P4, and P6, compared to the human subjects who did not use Interactive Human-aware Intervention. On the other hand, the median number of moves is lower for the human subjects who solved the planning tasks P10, P2, P7, and P9 using Interactive Human-aware Intervention. For the planning task P8, the medians are almost the same between those who used the Interactive Human-aware Intervention and those who did not.

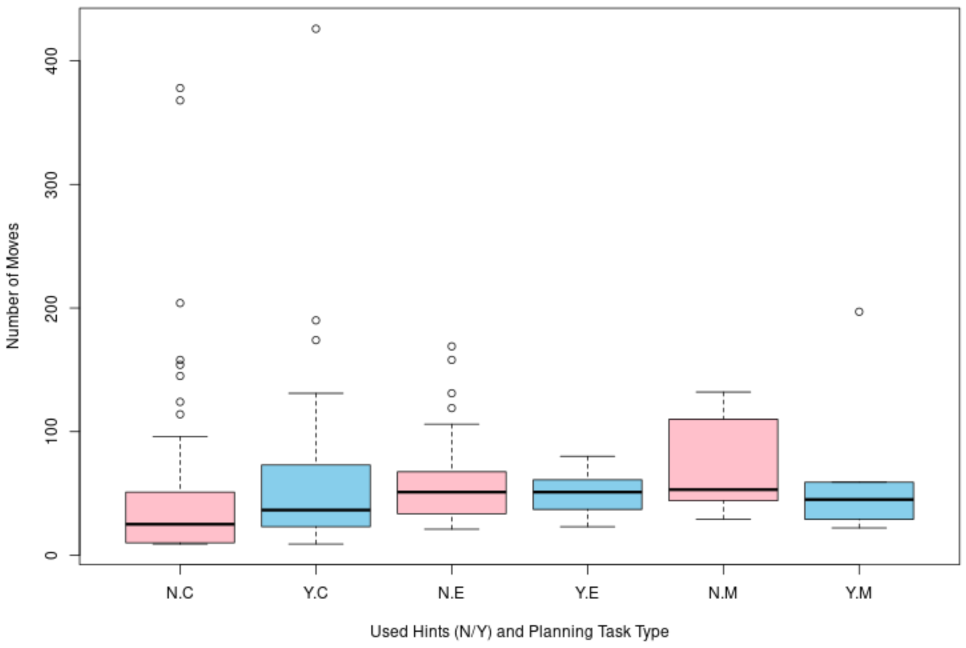


Figure 6.27: Box plots for the number of moves in the condition (Y) and the control group (N) for each planning task type

Figure 6.27 shows the effect on the number of moves with the number of moves and planning task type (C, E, M) as factors. Here too, the blue box plots represent the human subjects who used the Interactive Human-aware Intervention, while the red box plots represent the human subjects who did not. Using the Interactive Human-aware Intervention, human subjects who solved type M planning tasks have been able to reduce the median number of moves. In contrast, when using the Interactive Human-aware Intervention on type C puzzles the median number of moves have increased. For type E planning tasks, the median number of moves is almost the same between those who used the Interactive Human-aware Intervention and those who did not.

We verify whether or not the observed differences in the number of moves between the control and the condition group are statistically significant. Let μ_N be the mean number of moves in a solution from the control group (did not the Interactive Human-aware Intervention) and μ_H be the mean number of moves in a solution from the condition group (used the Interactive Human-aware Intervention). Then, we define the null (H_0) and the alternative (H_A) hypotheses as follows.

- $H_0 : \mu_N = \mu_H$, the means of the control and the condition groups are the same.
- $H_A : \mu_N \neq \mu_H$, the means of the control and the condition groups are different.

Throughout, we use two-way ANOVA with the type III correction to account for the unbalanced group design with $\alpha = 0.05$.

Considering the planning task id and the usage of the Interactive Human-aware Intervention as factors, ANOVA shows that the effect of the planning task id on the number of moves is significant ($p-value <<< 0, F = 14.0, df = 7$). However, the effect of the usage of the Interactive Human-aware Intervention is not significant on the number of moves ($p-value = 0.77, F = 0.08, df = 1$). The interaction effect of the usage of the Human-aware Intervention and the planning task id is also not significant ($p - value = 0.96, F = 0.28, df = 7$). However, the Levene's Test to test the homogeneity of variance fails ($p - value <<< 0, F = 4.27, df = 15$) when considering the factors planning task id and the usage of the Human-aware Intervention on the number of moves. The homogeneity of variance is a critical requirement for the ANOVA test. Furthermore the levels in the independent variable task id (P1, P2, ...) is associated with the levels of the independent variable type (C, E, M). For example, type C contains task ids P2, P4, P6, P8. Therefore, we can not report ANOVA considering task id as a factor. Because of these concerns, for the design that evaluates the effect on task id on the number of moves, we ignore the results and do not draw any conclusions from the analysis.

In contrast, considering the planning task type and the usage of the Interactive Human-aware Intervention as factors, the Levene's Test to test the homogeneity of variance passes ($p - value = 0.41, F = 1.0, df = 5$). Therefore, we report the conclusions from this design. The effect of the planning task type on the number of moves is not significant ($p-value = 0.62, F = 0.48, df = 2$). The effect of the usage of the Interactive Human-aware Intervention on the number of moves is not significant ($p - value = 0.61, F = 0.25, df = 1$). The interaction effect of the usage of the Human-aware Intervention and the planning task type on the number of moves is also not significant ($p - value = 0.74, F = 0.30, df = 2$) at $\alpha = 0.05$. Therefore, at $\alpha = 0.05$ we do not reject H_0 with respect to the usage of Interactive Human-aware Intervention and its effect on the number of moves.

Effect on the Difference from the Number of Moves in the Cost Optimal Solution

We address the question "*Does using the Interactive Human-aware Intervention have an effect on the difference from the number of moves in the optimal solution*"? This metric measures how close the human user's solution has got to the cost optimal solution for the planning task generated by an automated planner measured in the number of moves. We propose the following study design.

- **Dependent variable**:

  Let

  D_i = Difference from the cost optimal solution

  M_u = Number of moves in the human user's solution

  M_c = Number of moves in the cost optimal solution generated by a planner

$$D_i = M_u - M_c \tag{6.4}$$

- **Independent variables:**
  - Used the Interactive Human-aware Intervention (Yes / No)
  - Planning Task Id (P1, P2, P4, P6, P7, P8, P9, P10)
  - Planning Task Type (C, E, M)

Figure 6.28 illustrates the frequency probability distribution of the D_i in the solutions produced by the human users in the control and the condition groups. The data is highly skewed with a skewness factor +3.4. Because the sample size 181 is greater than the recommended size by the Central Limit Theorem, we assume that this is a normal distribution.

In Figure 6.29, the blue box plots represent the human subjects who used the Interactive Human-aware Intervention, while the red box plots represent the human subjects who did not.

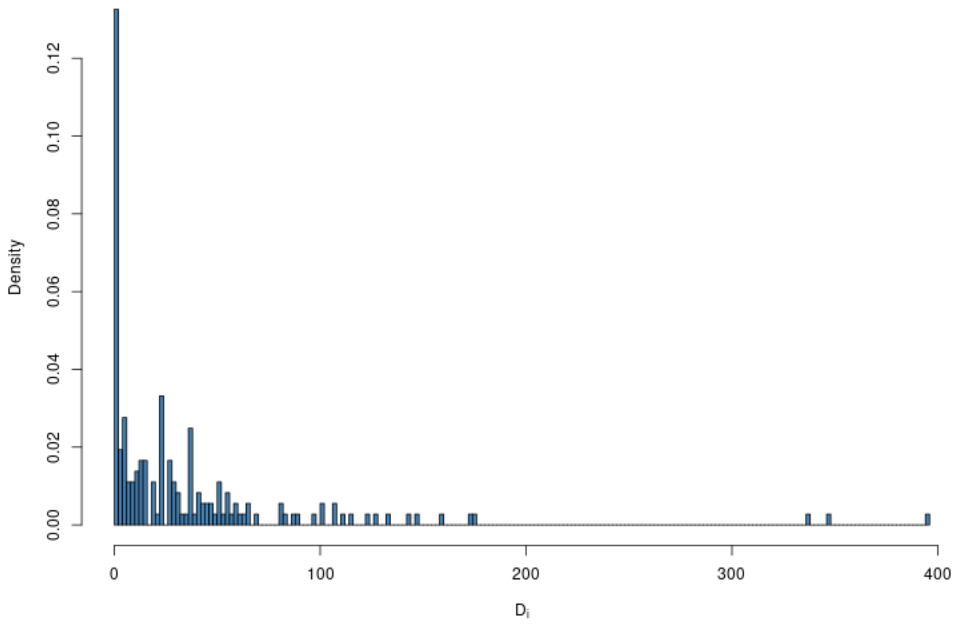


Figure 6.28: Frequency probability densities for the D_i distribution

Similar to the case where the number of moves was the dependent variable, the median D_i is higher for the human subjects who used the Interactive Human-aware Intervention for the planning tasks P1, P4, and P6. The median D_i is lower for human subjects who solved the planning tasks P10, P2, P7, and P9 using the Interactive Human-aware Intervention. For the planning task P8, there seems to be no difference between the D_i medians of the two groups.

Figure 6.30 shows the effect on the dependent variable D_i by planning task type (C, E, M). Using the Interactive Human-aware Intervention, human subjects who solved type M planning tasks have been able to reduce the median D_i. In contrast, when using the Interactive Human-aware Intervention on type C planning task, the median D_i has increased. There seems to be no difference between the D_i medians of those two solved type E planning tasks using the Interactive Human-aware Intervention and those who did not use the Interactive Human-aware Intervention.

We verify whether or not the observations for dependent variable between the control and the condition group are statistically significant. Let μ_N be the mean difference to the optimal (i.e., mean of D_i) in the solutions from the control group which did not the Interactive Human-aware

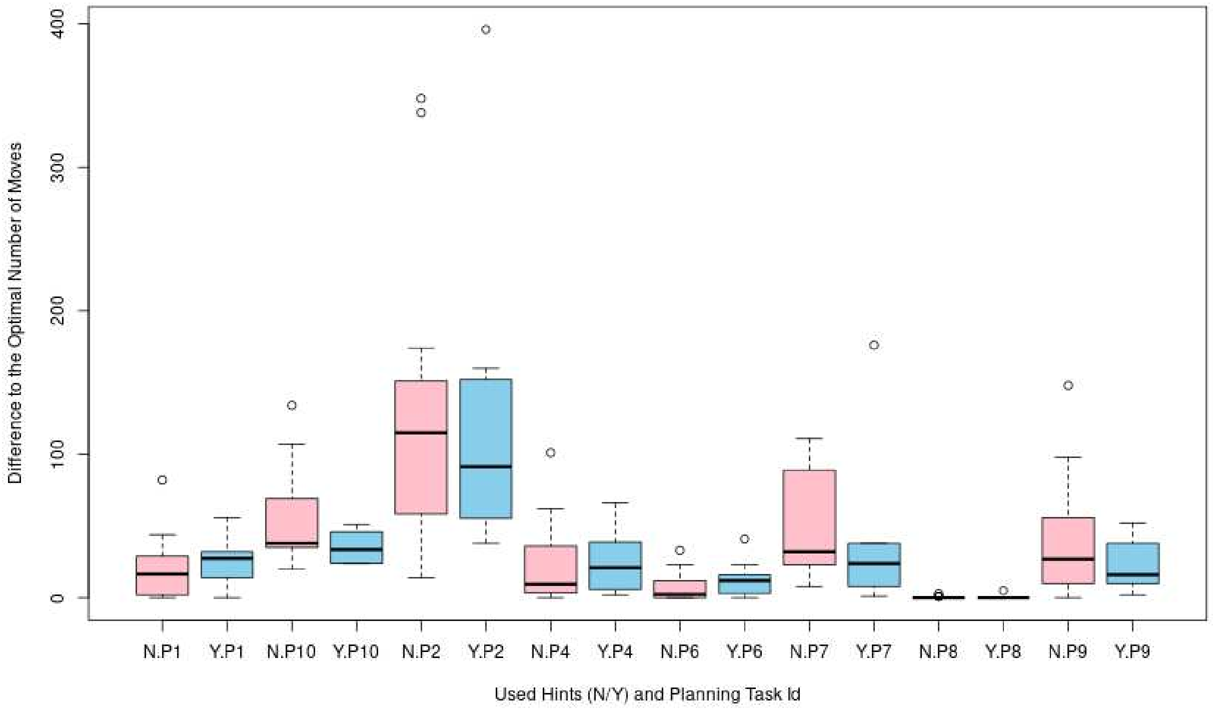


Figure 6.29: Box plot for the difference from the optimal in the condition and the control group for each planning task id

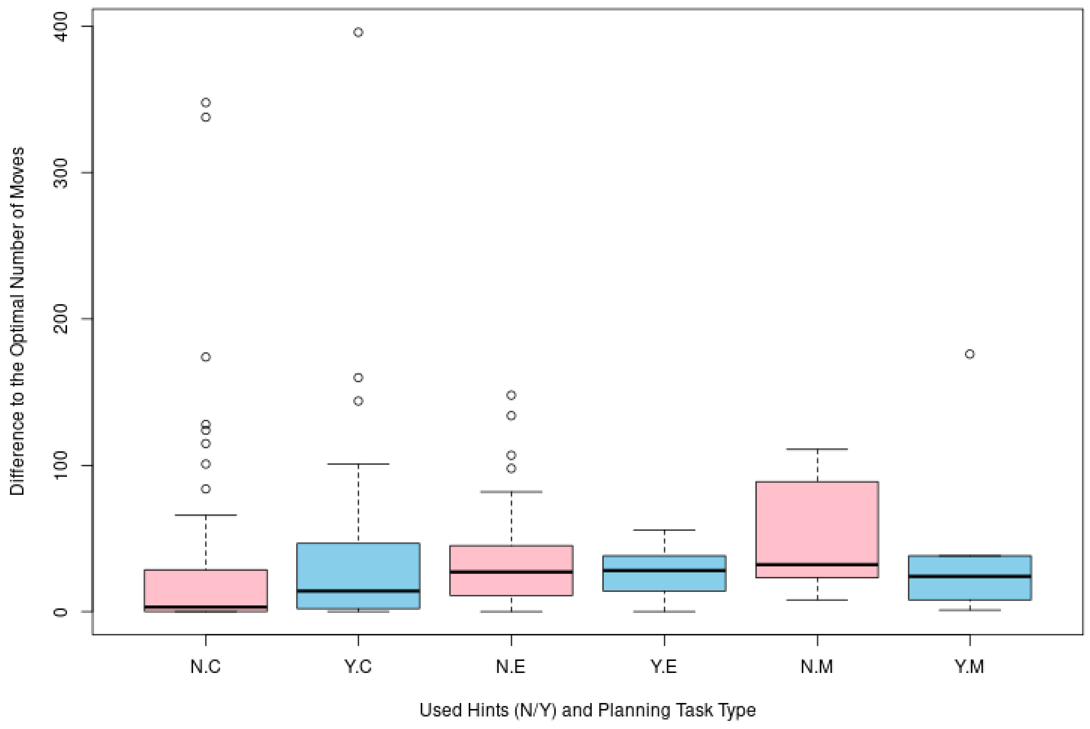


Figure 6.30: Box plot for the difference from the optimal in the condition and the control group for each planning task type

Intervention) and μ_H be the mean difference to the optimal in a solution (i.e., mean of D_i) from the condition group which used the Interactive Human-aware Intervention. Then, we define the null (H_0) and the alternative (H_A) hypotheses as follows.

- $H_0 : \mu_N = \mu_H$, the means of the control and the condition group are the same.
- $H_A : \mu_N \neq \mu_H$, the means of the control and the condition group are different.

Throughout, we use two-way ANOVA with the type III correction to account for the unbalanced group design with $\alpha = 0.05$.

As in the case where the dependent variable is the number of moves, the Levene's Test to test the homogeneity of variance fails ($p - value <<< 0, F = 4.3, df = 15$) for the design where we consider the planning task id and the usage of the Interactive Human-aware Intervention as factors. Therefore, we ignore the results and do not draw any conclusions from the ANOVA.

In contrast, considering the planning task type and the usage of the Interactive Human-aware Intervention as factors, the Levene's Test to test the homogeneity of variance passes ($p - value = 0.69, F = 0.62, df = 5$). Therefore, we report the conclusions from this design. The effect of the planning task type on D_i is not significant ($p - value = 0.68, F = 0.17, df = 2$). The effect of the usage of the Interactive Human-aware Intervention on the differences to the optimal is not significant ($p - value = 0.63, F = 0.46, df = 1$). The interaction effect of the usage of the Human-aware Intervention and the planning task type on D_i is also not significant ($p - value = 0.76, F = 0.27, df = 2$) at $\alpha = 0.05$. Therefore, at $\alpha = 0.05$ we do not reject H_0 with respect to the usage of Interactive Human-aware Intervention and its effect on the solution length differences to the optimal solution produced by an automated planner.

Computing the Evaluation Metric and the Data Preparation for Evaluation Question 2

We use Figure 6.31 as an example to compute the evaluation metric, the latest time the landmarks are achieved. In Figure 6.31, there are three plans for the same planning task produced by an automated planner, user X and user Y respectively. The plans are drawn in proportion to the number of steps. The plan produced by the automated planner has 10 steps. User X has produced

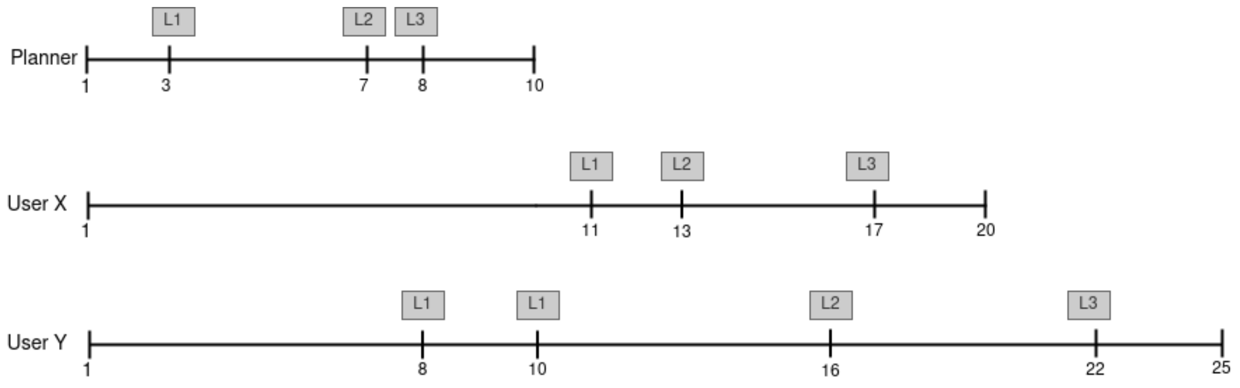


Figure 6.31: Plans produced by an automated planner (top), User X (middle) and User Y (bottom) achieving the same three landmarks.

Table 6.14: Latest Landmark Achievement Times

Plan	Landmark		
	L1	L2	L3
Planner	3	7	8
User X	11	13	17
User Y	10	16	22

a 20 step plan. User Y has produced 25 step plan. Let us assume L1, L2 and L3 are the landmarks for the planning task.

Table 6.14 shows the latest time in number of steps each landmark is achieved. Note that User Y has initially achieved the L1 at step 8 but has lost and regained it at step 10. Therefore the latest landmark achievement time for L1 for User Y is 10. We compute the landmark achievement metric $\mathcal{L}_A$ as follows:

Let

L_i^u = Latest time the user achieved the landmark L_i in number of steps

L_i^p = Latest time the automated planner achieved the landmark L_i in number of steps

$$\mathcal{L}_A = \frac{\sum_{L_i \in Landmarks} |(L_i^u - L_i^p)|}{|Landmarks|} \tag{6.5}$$

For the example in Figure 6.31:

- landmark achievement for User X $= |(11 - 3)| + |(13 - 7)| + |(17 - 8)| = 23/3 \approx 8steps$

- landmark achievement for user Y $= |(10-3)| + |(16-7)| + |(22-8)| = 30/3 = 10 steps$

Note that we compute the landmark achievement averaged over the number of landmarks in the planning task. This is because different planning tasks used in this study have different number of landmarks. In Section 6.7.6 we saw that the human users who achieve the landmarks for a planning task similar to the automated planner, find the solution to the planning task faster compared to human users who do not. The landmark achievement metric captures this intuition. The closer the user's solution is to the solution of the planner, the smaller the sum of the differences of achievement will be. Thus, the metric indicates whether the user is getting closer to the optimal solution to the planning task or not. Therefore, we use the landmark achievement metric to evaluate whether the Interactive Human-aware Intervention is helpful in moving the human user closer to the optimal solution produced by a planner.

Figure 6.32 illustrates the frequency probability distribution of the landmark achievement in the solutions produced by the human users in the control and the condition groups. The data is highly skewed with a skewness factor +4.2. Given the sample size 181 being greater than the recommended size by the Central Limit Theorem, we assume that this is a normal distribution.

Effect on the Latest Time the Landmarks Are Achieved

We address the question "Does using the Interactive Human-aware Intervention have an effect on the landmark achievement?" We propose the following study design.

- **Dependent variable**: landmark achievement ($\mathcal{L}_A$)
- **Independent variables:**
  - Used the Interactive Human-aware Intervention (Yes / No)
  - Planning Task Id (P1, P2, P4, P6, P7, P8, P9, P10)
  - Planning Task Type (C, E, M)

In Figure 6.33, the blue box plots represent the human subjects using the Interactive Human-aware Intervention, while the red box plots represent the human subjects who do not. The median

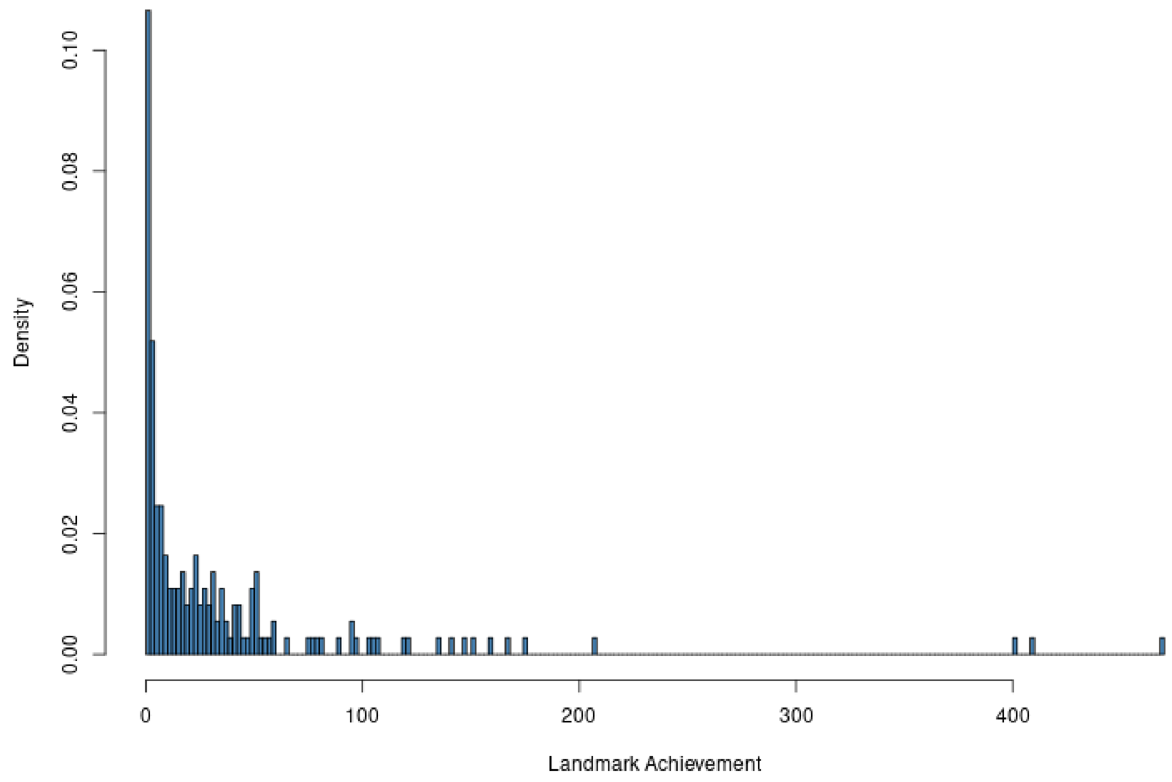


Figure 6.32: Frequency probability distribution of the landmark achievement

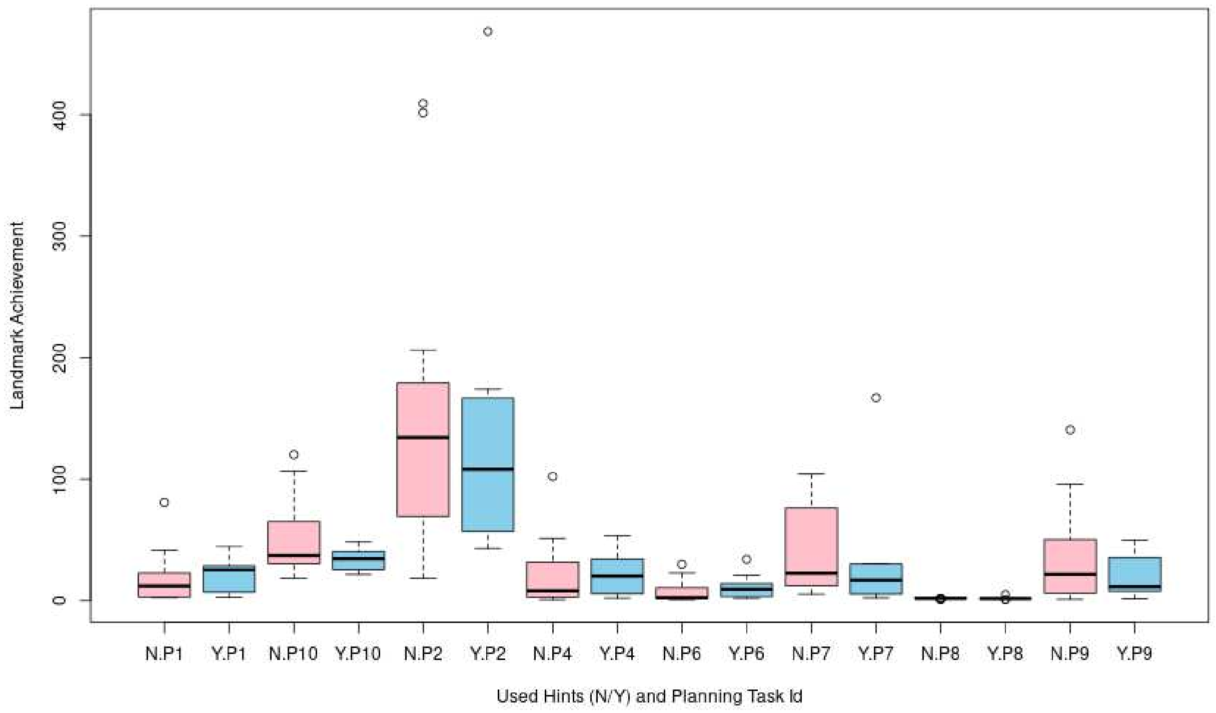


Figure 6.33: Box plot for the landmark achievement in the condition and the control group for each planning task id

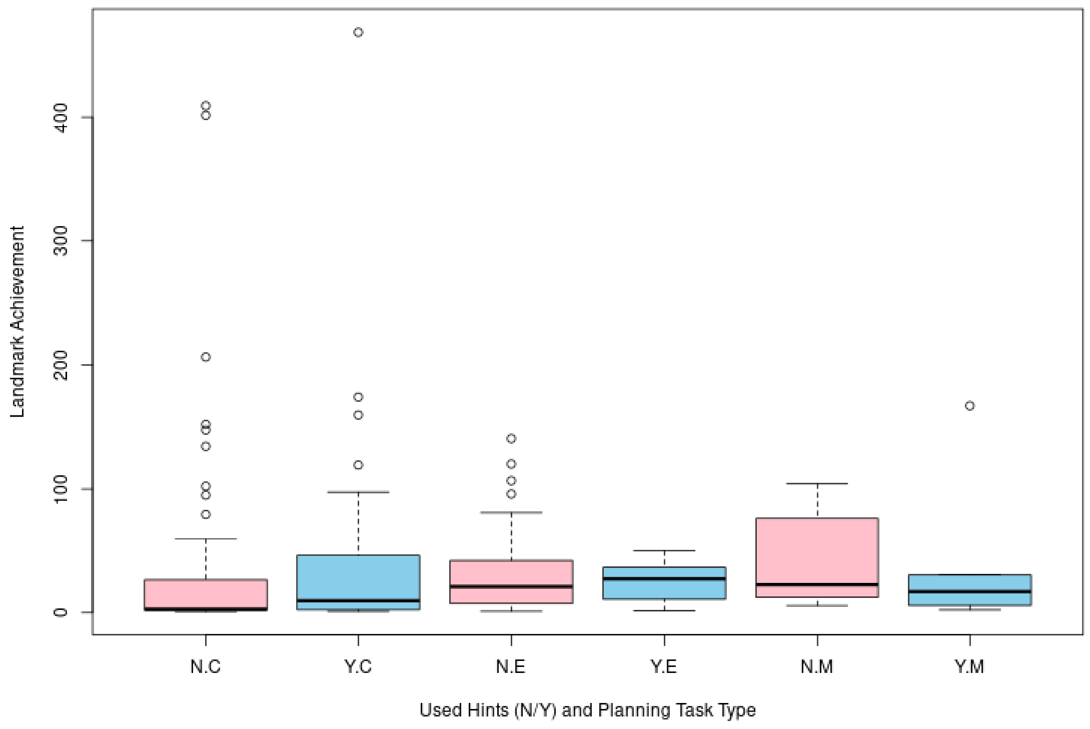


Figure 6.34: Box plot for the landmark achievement in the condition and the control group for each planning task type

$\mathcal{L}_A$ for the human subjects using the Interactive Human-aware Intervention for the planning tasks P1, P4, and P6 are higher. The human subjects who solved the planning tasks P2, P7, P9, P10 with the Interactive Human-aware Intervention reported lower median landmark achievement. For the planning task P8, there seems to be no difference in the median $\mathcal{L}_A$ when using Interactive Human-aware Intervention compared to not using Interactive Human-aware Intervention.

Figure 6.34 shows the effect on the dependent variable by planning task type (C, E, M). Using the Interactive Human-aware Intervention, human subjects who solved type M planning tasks have been able to reduce the median $\mathcal{L}_A$. In contrast, when using the Interactive Human-aware Intervention on type E and C planning task, the median $\mathcal{L}_A$ have increased.

We verify whether or not the observations for dependent variable between the control and the condition group are statistically significant. Let μ_N be the mean $\mathcal{L}_A$ in a solution from the control group (did not the Interactive Human-aware Intervention) and μ_H be the mean $\mathcal{L}_A$ in a solution from the condition group (used the Interactive Human-aware Intervention). Then, we define the null (H_0) and the alternative (H_A) hypotheses as follows.

- $H_0 : \mu_N = \mu_H$, the mean $\mathcal{L}_A$ values of the control and the condition group are the same.
- $H_A : \mu_N \neq \mu_H$, the mean $\mathcal{L}_A$ values of the control and the condition group are different.

Throughout, we use two-way ANOVA with the type III correction to account for the unbalanced group design with $\alpha = 0.05$.

When using Interactive Human-aware Intervention and the planning task id as factors of the dependent variable $\mathcal{L}_A$, the Levene's Test to test the homogeneity of variance fails ($p-value <<< 0, F = 5.08, df = 15$) for the design. Furthermore the levels in the independent variable task id (P1, P2, ...) is associated with the levels of the independent variable type (C, E, M). For example, type C contains task ids P2, P4, P6, P8. Therefore, we can not report ANOVA considering task id as a factor. Because of these reasons, we ignore the results and do not draw any conclusions from the ANOVA.

In contrast, considering the planning task type and the usage of the Interactive Human-aware Intervention as factors, the Levene's Test to test the homogeneity of variance passes ($p - value = 0.65, F = 0.67, df = 5$). Therefore, we report the conclusions from this design. The effect of the planning task type on the $\mathcal{L}_A$ is not significant ($p - value = 0.86, F = 0.15, df = 2$). The effect of the usage of the Interactive Human-aware Intervention on the $\mathcal{L}_A$ is also not significant ($p - value = 0.83, F = 0.05, df = 1$) at $\alpha = 0.05$. The interaction effect of the usage of the Human-aware Intervention and the planning task type on the landmark achievement is not significant ($p - value = 0.91, F = 0.09, df = 2$). Therefore, at $\alpha = 0.05$ we do not reject H_0 with respect to $\mathcal{L}_A$ for the design that considers the factors usage of Interactive Human-aware Intervention and the planning task type.

Computing the Evaluation Metric and the Data Preparation for Evaluation Question 2

We use the example illustrated in Figure 6.31 to compute the evaluation metric, the number of times the landmarks are regained. In the example, the planner has achieved each landmark L1, L2, L3 only once. User X has also achieved each landmark only once. User Y has achieved the landmark L1 twice: initially achieved the at step 8 but has lost and regained it at step 10. Therefore, Y has regained the landmark once. Additionally Y has achieved the landmarks L2 and L3 only once. We compute the landmark regain metric ($\mathcal{L}_R$) as follows:

$$\mathcal{L}_R = \sum_{Li \in Landmarks} |(\text{number of times } Li \text{ appears in the solution} - 1)| \tag{6.6}$$

For the example in Figure 6.31:

- $\mathcal{L}_R$ for User X $= |(1-1)| + |(1-1)| + |(1-1)| = 0$
- $\mathcal{L}_R$ for user Y $= |(2-1)| + |(1-1)| + |(1-1)| = 1$

In Section 6.7.6 we saw that the human users who achieve the landmarks for a planning task similar to the automated planner, find the solution to the planning task faster compared to human users who do not. The landmark achievement metric captures this intuition. Furthermore, for the Rush Hour planning tasks the cost optimal solutions achieve each landmark only once in most cases. When the user gains a landmark and loses it by executing another action, it indicates that the user is moving away from the solution. The closer the user's solution is to the solution of the planner, the smaller the landmark regain value will be. Thus, the metric indicates whether the user is getting closer to the optimal solution to the planning task or not. Therefore, we use the landmark regain metric to evaluate whether the Interactive Human-aware Intervention is helpful in moving the human user closer to the optimal solution.

The control group has 113 observations. The condition group has 68 observations. Figure 6.35 illustrates the frequency probability distribution of the landmark achievement in the solutions produced by the human users in the control and the condition groups. The data is highly skewed with a skewness factor +3.5.

There are many zeros in the data set, indicating that many subjects achieved the landmarks for their planning problem only once. Given the sample size 181 being greater than the recommended size by the Central Limit Theorem, we assume that this is a normal distribution.

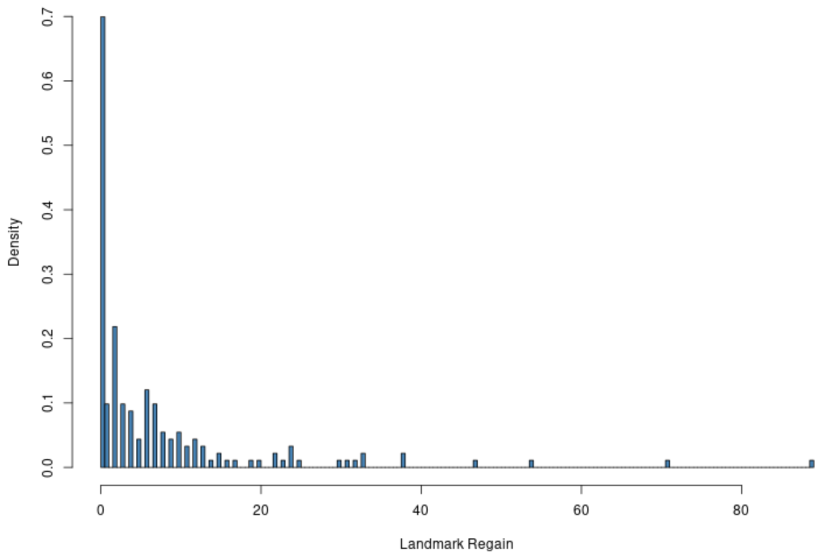


Figure 6.35: Frequency probability distribution of the landmark regain

Effect on Regaining Landmarks

We address the question "Does using the Interactive Human-aware Intervention have an effect on the landmark regain?" We propose the following study design.

- **Dependent variable**: landmark regain ($\mathcal{L}_R$)
- **Independent variables:**
  - Used the Interactive Human-aware Intervention (Yes / No)
  - Planning Task Id (P1, P2, P4, P6, P7, P8, P9, P10)
  - Planning Task Type (C, E, M)

In Figure 6.36, the blue box plots represent the human subjects using the Interactive Human-aware Intervention, while the red box plots represent the human subjects who do not. The median $\mathcal{L}_R$ for the human subjects using the Interactive Human-aware Intervention for the planning tasks P1, P4, P6 and P7 is higher. The median $\mathcal{L}_R$ for the human subjects using the Human-aware Intervention for the planning tasks P2 and P9 is lower. The medians $\mathcal{L}_R$ for P8 and P10 are almost the same.

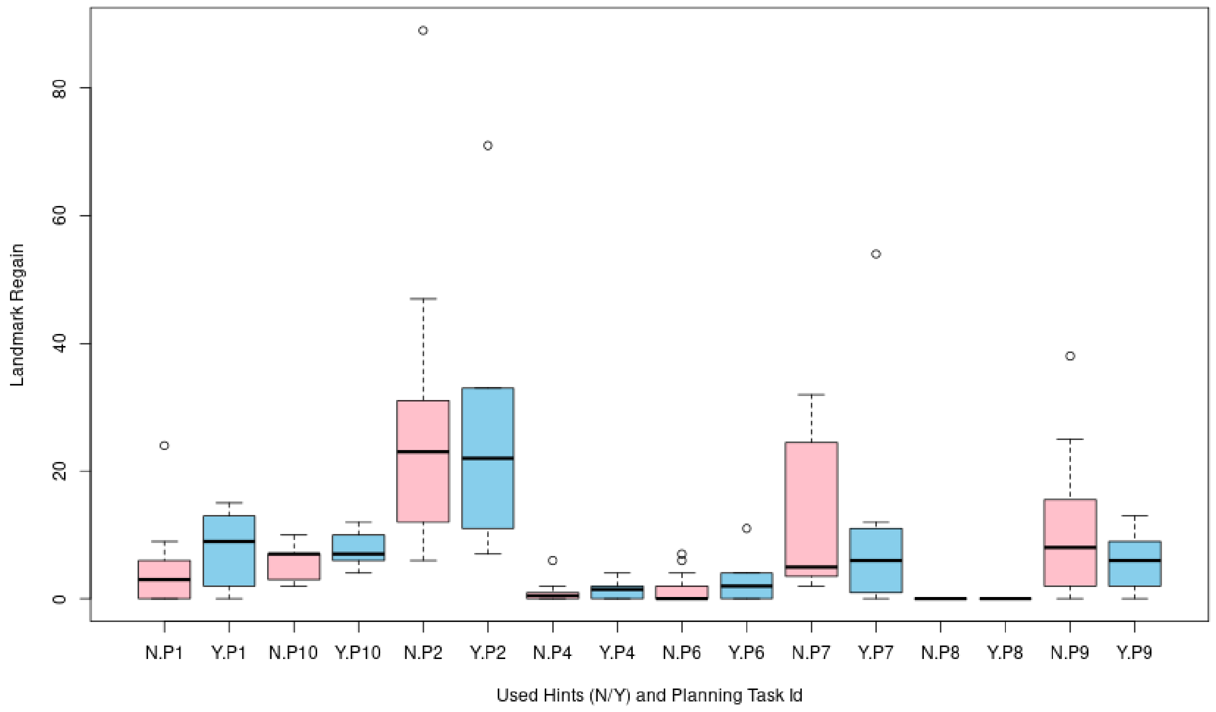


Figure 6.36: Box plot for the landmark regain in the condition and the control group for each planning task id

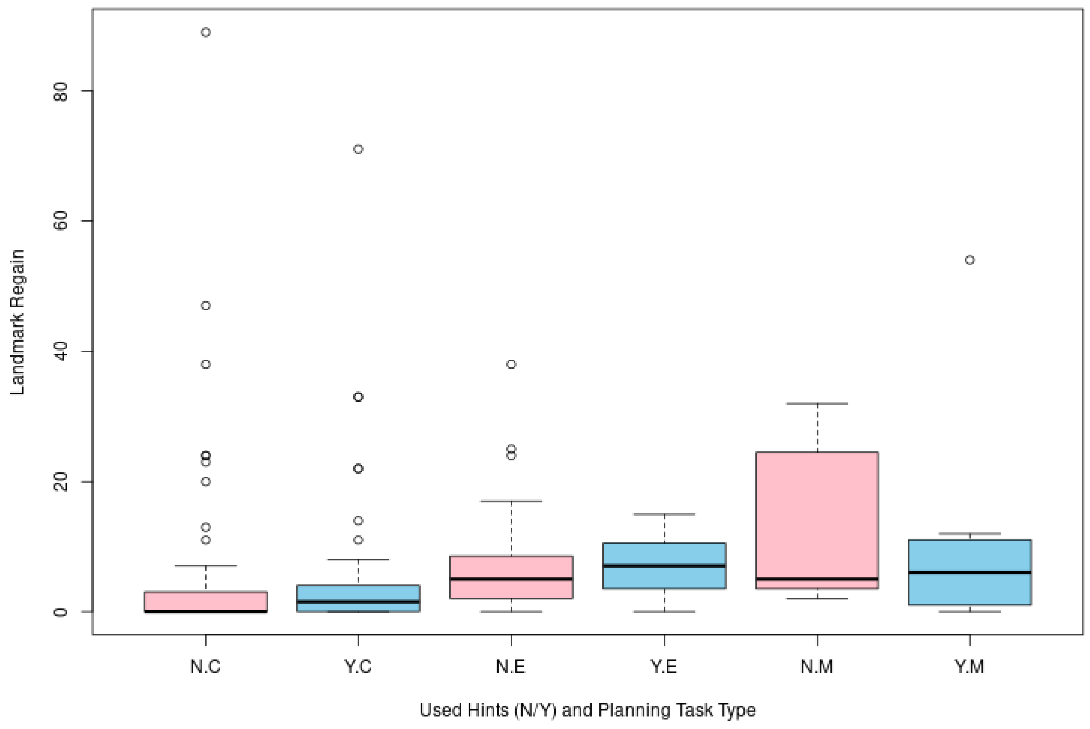


Figure 6.37: Box plot for the landmark regain in the condition and the control group for each planning task type

Figure 6.37 shows the effect on the dependent variable by planning task type (C, E, M). When using the Interactive Human-aware Intervention on type E, M and C planning task, the median $\mathcal{L}_R$ values have increased.

We verify whether or not the observations for the dependent variable $\mathcal{L}_R$ differences between the control and the condition group are statistically significant. Let μ_N be the mean $\mathcal{L}_R$ in a solution from the control group (did not the Interactive Human-aware Intervention) and μ_H be the mean $\mathcal{L}_R$ in a solution from the condition group (used the Interactive Human-aware Intervention). Then, we define the null (H_0) and the alternative (H_A) hypotheses as follows.

- $H_0 : \mu_N = \mu_H$, the means of the control and the condition group are the same.
- $H_A : \mu_N \neq \mu_H$, the means of the control and the condition group are different.

Throughout, we use two-way ANOVA with the type III correction to account for the unbalanced group design with $\alpha = 0.05$.

When using Interactive Human-aware Intervention and the planning task id as factors to the dependent variable landmark achievement, the Levene's Test to test the homogeneity of variance fails ($p - value <<< 0, F = 4.3, df = 15$) for the design. The Levene's test passes for the design that considers the usage of Interactive Human-aware Intervention and the planning task type ($p - value = 0.62, F = 0.71, df = 5$). Therefore, we only report the ANOVA results from the design that considers that considers the usage of Interactive Human-aware Intervention and the planning task type.

The effect of the planning task type on the $\mathcal{L}_R$ is not significant ($p - value = 0.18, F = 1.7, df = 2$). The effect on the usage of Interactive Human-aware Intervention on the $\mathcal{L}_R$ is not significant ($p - value = 0.8, F = 0.06, df = 1$). The interaction effect of the usafe of Interactive Human-aware Intervention and the planning task type on $\mathcal{L}_R$ is also not significant ($p - value = 0.96, F = 0.03, df = 2$). Therefore, at $\alpha = 0.05$ we do not reject H_0 with respect to $\mathcal{L}_R$ for the design that considers the factors, the usage of Interactive Human-aware Intervention and the planning task type.

Effect of Help Video

We address the question "Does seeing a help video move the user closer to the optimal solution?". For the analysis we use the data from the Interactive Human-aware Intervention human subject study, where 135 participants completed 13 planning tasks. Recall that during the experiment the participants were randomly assigned to two groups where one group watched a 44 second help video on how to solve the Rush Hour planning task while avoiding the forbidden vehicle. The other group did not watch the help video. 72 subjects watched the help video, while 63 subjects did not. We propose the following study design.

- **Dependent variable**: Number of moves
- **Independent variables:**

  - Used the Interactive Human-aware Intervention (Yes / No)
  - Watched the help video (Yes/No)
  - Planning Task Id (P1 through P13)
  - Planning Task Type (C, E, M)

We join the two variables: **using the Interactive Human-aware Intervention** and **watching the help video** to form one independent variable with 4 levels. Level 1, Yes-Yes (YY) indicates that the participant watched the video and used the Interactive Human-aware Intervention. Level 2, Yes-No (YN) indicates that the participant watched the help video but did not use the Interactive Human-aware Intervention. Level 3, No-Yes (NY) indicates that the participant did not watch the help video but used the Interactive Human-aware Intervention. Level 4, No-No (NN) indicates that the participant did not watch the video or used the Interactive Human-aware Intervention. We use this combined variable called **Help Type** for the analysis that follow. There are 46 users in level 1, 26 in level 2, 37 in level 3 and 26 in level 4. Thus, our modified design is as follows:

- **Dependent variable**: Number of moves
- **Independent variables:**
  - Help Type (YY/NY/YN/NN)
  - Planning Task Id (P1 through P13)
  - Planning Task Type (C, E, M)

Figure 6.38 illustrates the frequency probability distribution of the number of moves for the solutions produced by the human users in the sample. The data is highly skewed with a skewness factor +4.6.

The frequency distribution in the sample for the planning tasks and each category of help type is shown in Table 6.15.

In Figure 6.39, the box plots represent the human subjects using the four help types for the 13 planning tasks. Table 6.16 summarizes the median number of moves for the thirteen planning

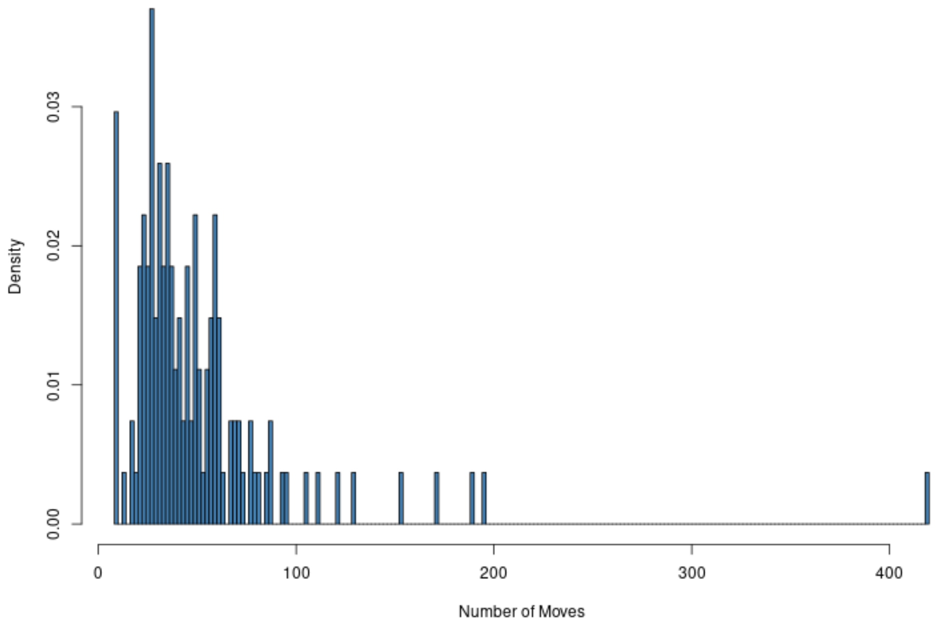


Figure 6.38: Frequency probability distribution of the number of moves

Table 6.15: Frequency distribution in the sample for the planning task id and the types of help

Planning	Help Type			
Task	YY	YN	NY	NN
P1	-	1	3	2
P2	3	1	4	-
P3	8	-	5	-
P4	6	2	3	3
P5	8	-	6	1
P6	1	3	2	3
P7	6	-	-	1
P8	-	6	1	2
P9	2	3	1	3
P10	3	-	5	-
P11	2	4	2	5
P12		5	-	5
P13	7	1	5	1

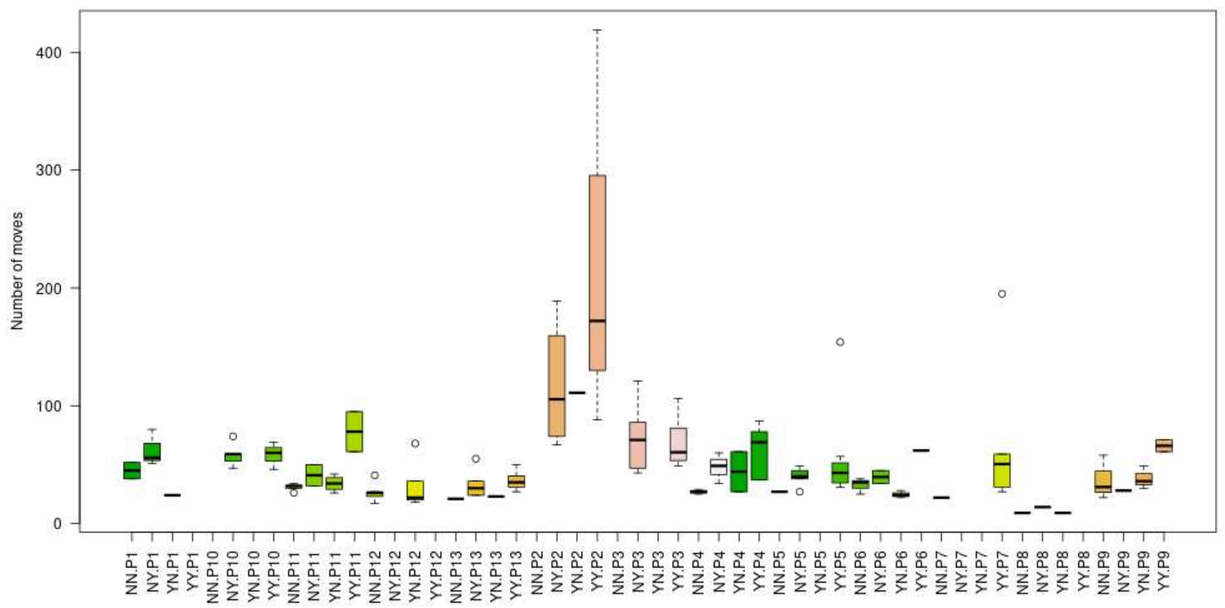


Figure 6.39: Box plot for the number of moves in the sample for the planning task id and the help types

tasks grouped by the type of help received by the human subjects attempting each task. Within parenthesis are the mean number of moves. Recall that as per Table 6.15, there was only one participant assigned to some help configuration. We have marked such occurrences with an asterisk (*) in Table 6.16 and they are omitted from the discussion that follows. For the planning task P1, the median number of moves for the subjects who used the Interactive Human-aware Intervention (NY) is higher than the subjects who did not use any kind of help. For the planning task P2, the median number of moves for the subjects receiving both kinds of help is higher than those who only watched the help video. For the planning task P3, the median number of moves for the subjects receiving both kinds of help solved the planning task faster than the subjects who only used the Interactive Human-aware Intervention. For the planning task P4, the median number of moves is lowest for subjects who did not use either kind of help. Furthermore, the subjects who received both kinds of help reported the highest median number of moves. When comparing subjects who used only one type of help, those who only watched the help video reported a lower median number of moves compared to those who used the Interactive Human-aware Intervention. For the planning task P5, the median number of moves for the subjects who only used the Interactive Human-aware Intervention is lower than the subjects who used both kinds of help. For the planning task P6, the subjects who only watched the help video reported the lowest median number of moves compared to those who only used the Interactive Human-aware Intervention and those who did not use any kind of help. For the planning task P7, there is one subject who used no help, while all others who solved P7 used both types of help. For the planning task P8, the median number of moves between the subjects who only watched the help video and the subjects who did not use any kind of help are the same. For the planning task P9, the subjects receiving both kinds of help reported the highest median number of moves compared to the subjects who only watched the help video or subjects who did not use any kind of help. However, the median number of moves for subjects who used the Interactive Human-aware Intervention is lower compared to subjects who did not use any kind of help. For the planning task P10, the subjects receiving both kinds of help reported a slightly higher median number of moves compared to the subjects who only used the Interactive Human-aware Intervention. However, the mean number of moves between the two groups are the same. For the planning task P11, the subjects receiving both kinds of help reported the highest median number of moves compared to all other help types. The lowest median number of moves was reported for subjects who did not use any kind of help. When comparing subjects who used only one type of help, those who only watched the help video reported a lower median number of moves compared to those who only used the Interactive Human-aware Intervention. For the planning task P12, the subjects who watched the help video reported a lower median number of moves compared to the subjects who did not use any kind of help. However, the mean number of moves for the subjected who watched the help video was higher compared to the subjects who did not use any kind of help. For the planning task P13, the subjects receiving both kinds of help reported a higher median number of moves compared to the subjects who only used the Interactive Human-aware Intervention.

In summary, we see that considering the median number of moves, for many planning tasks used in this study, using both Interactive Human-aware Intervention and the Help video has resulted

Table 6.16: The median and the mean (in parenthesis) number of moves for the planning tasks P1 through P13 for the different types of help received by the human subjects. YY, YN, NY, NN are the help types. YY means the subject used both the Human-aware Intervention and watched the help video. YN means that the subject watched the help video but did not use the Interactive Human-aware Intervention. NY means that the subject did not watch the help video but used the Interactive Human-aware Intervention. NN means that the subject did not watch the help video or use the Interactive Human-aware Intervention.

Planning Task	Number of Moves median (mean)			
	YY	YN	NY	NN
P1		24 (24)*	56 (62)	45 (45)
P2	172 (226)	111 (111)*	105 (117)	
P3	60 (68)		74 (71)	
P4	69 (63)	44 (44)	49 (48)	27 (27)
P5	43 (55)		40 (40)	27 (27)*
P6	62 (62)*	24 (25)	39 (39)	35 (33)
P7	50 (69)			22 (22)*
P8		9 (9)	14 (14)*	9 (9)
P9	66 (66)	36 (36)	28 (28)*	31 (31)
P10	60 (58)		59 (58)	
P11	78 (78)	34 (34)	41 (41)	32 (31)
P12		22 (33)		26 (27)
P13	35 (36)	23 (23)*	30 (34)	21 (21)*

in increasing the median number of moves in the solutions produced by the human users (e.g., P2, P4, P5, P7, P9, P11, P13). For the planning tasks P3, using both help types reduced the median number of moves. For the planning task P10, using both help types only slightly increased the median number of moves, but the mean remained the same. Interestingly, for several planning tasks (e.g., P1, P4, P9, P11) human users found shorter solutions when they did not use any kind of help.

Figure 6.40 shows the effect on the number of moves by planning task type (C, E, M) and the type of help received by the participants. We summarize the median number of moves for each planning task type in Table 6.17. Within parenthesis are the mean number of moves. For type C planning tasks we see that the median number of moves for those receiving both kinds of help is the highest compared to the other three help types. Furthermore, the second highest median is reported for subjects who only used the Interactive Human-aware Intervention. The lowest median number of moves is reported for subjects who did not use any kind of help. The mean number of

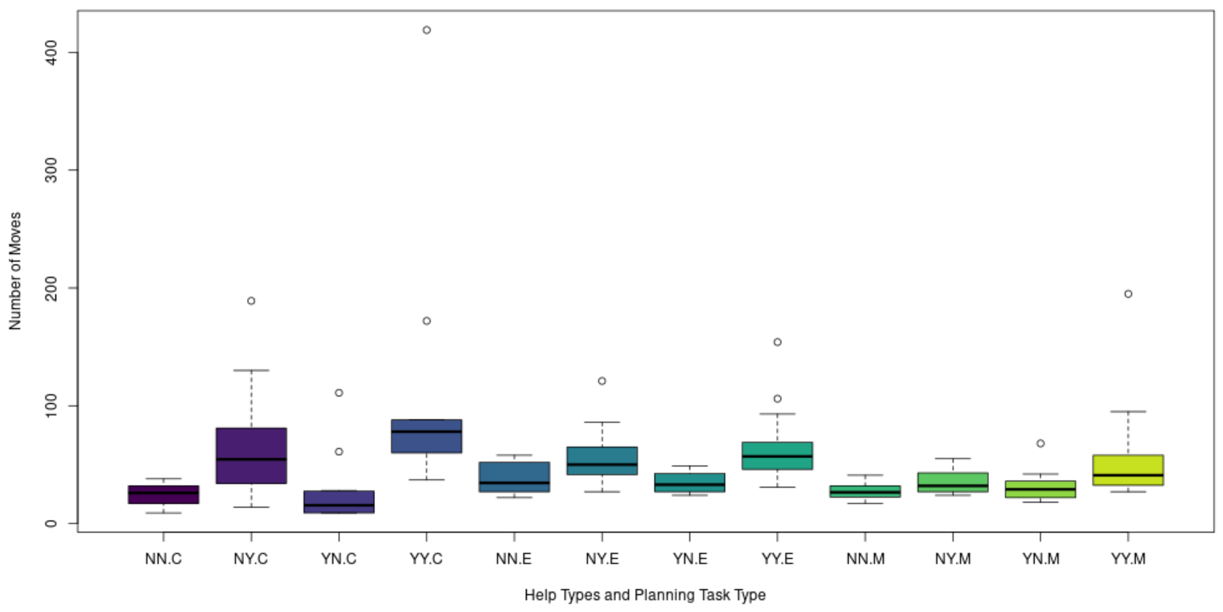


Figure 6.40: Box plot for the number of moves in the sample for each planning task type and the type of help they received

moves for type C planning tasks follow the same pattern for the highest and the lowest means in the category. For type E planning tasks, the lowest median and the mean is reported for subjects who only watched the help video. The highest mean and the median number of moves are reported for subjects who received both kinds of help. For type E planning tasks, among those who used Interactive Human-aware Intervention, lower median number of moves is reported for those who only used the Interactive Human-aware Intervention. For type M planning tasks, the lowest median and the mean number of moves are reported for subjects who did not use any type of help. The highest median and the mean number of moves are reported for those who used both kinds of help. Similar to type E planning tasks, among those who used Interactive Human-aware Intervention, lower median and mean is reported for those who did not watch the help video.

We verify whether or not the differences in the number of moves between the help types and the planning task types are statistically significant. We define the null (H_0) and the alternative (H_A) hypotheses as follows:

- H_0: there is no difference in the mean number of moves between the help types.

Table 6.17: The median and the mean (in parenthesis) number of moves for the planning tasks types C, E, M for the different types of help received by the human subjects. YY, YN, NY, NN are the help types. YY means the subject used both the Human-aware Intervention and watched the help video. YN means that the subject watched the help video but did not use the Interactive Human-aware Intervention. NY means that the subject did not watch the help video but used the Interactive Human-aware Intervention. NN means that the subject did not watch the help video or use the Interactive Human-aware Intervention. C, E, M refers to the planning task categories based on the position of the forbidden vehicle. C means that the forbidden vehicle is in the **c**orner or the board. E means that the forbidden vehicle is placed in an **e**dge of the board. M means that the forbidden vehicle is placed in the **m**iddle of the board.

Planning Task Type	Number of Moves median (mean)			
	YY	YN	NY	NN
C	78 (112)	15 (27)	54 (70)	26 (25)
E	57 (62)	33 (35)	50 (56)	34 (38)
M	41 (55)	29 (32)	32 (36)	26 (28)

- H_A :, there is a difference in the mean number of moves between the help types.

We use two-way ANOVA with $\alpha = 0.05$. We do not consider the planning task id as a factor because there were no subjects for certain help categories, while for several other help categories we only had only one or two subjects. Therefore, our analysis on the effect of using the four help types for the three planning task types uses the following study design.

- **Dependent variable**: Number of moves
- **Independent variables:**
  - Help Type (YY/NY/YN/NN)
  - Planning Task Type (C, E, M)

We conducted the Levene's Test to test the homogeneity of variance for this design. The test passes for this design ($p - value = 0.09, F = 1.65, df = 11$) indicating that we can use the two-way ANOVA to draw conclusions from the design. The results show that the effect on the mean number of moves considering the help type factor is significant ($p - value <<< 0, F = 8.2, df = 3$). The effect on the mean number of moves considering the planning task type factor is not significant ($p - value = 0.07, F = 2.8, df = 2$) The effect on the mean number of moves considering the interaction effect of the planning task type and the help type is not significant ($p - value = 0.09, F = 1.8, df = 6$)

Since help type significantly affects the number of moves in a solution, we want to find out for which specific individual planning task types using the help type reduces/increases the number of moves. For example, is the difference in mean number of moves for YY and NY for planning task type C significant? Is the difference in mean for YY and NY for planning task M significant? and so on. We address this question by conducting the TukeyHSD posthoc analysis to compare the pair-wise means. For simplicity, we only discuss the pairs that returned statistically significant differences of means. For type C planning tasks the differences of means between YY and NN help types is statistically significant ($p - value = 0.001$). This means that for type C planning tasks when human users use both kinds of help, the number of moves increases compared to instances where they do not use any kind of help, and that this increase is statistically significant. For type C planning tasks the differences of means between YY and YN is also statistically significant ($p - value = 0.0003$). This means that for type C planning tasks when human users have already watched the help video, using the Interactive Human-aware Intervention increases the solution length compared to instances where they do not use the Interactive Human-aware Intervention. Considering type E and C planning tasks, the mean number of moves between type C planning task when both types of help is used is higher compared to type E planning task when using no help. Posthoc analysis showed that this difference is statistically significant ($p - value = 0.03$). The mean number of moves between type C planning task when both types of help is used is significantly higher compared to type E planning task when only using Interactive Human-aware Intervention as the help type ($p - value = 0.03$). Comparing type M and type C planning tasks, the mean number of moves between type C planning task when both types of help is used is significantly higher compared to type M planning task when using no help ($p - value = 0.0003$). The mean number of moves between type C planning task when both types of help is used is significantly higher compared to type M planning task when only using Interactive Human-aware Intervention as the help type ($p - value = 0.01$). The mean number of moves between type C planning task when both types of help is used is significantly higher compared to type M planning task when only using the help video as the help type ($p - value = 0.001$). The mean number of moves between type C planning task when both types of help is used is significantly higher compared to type M planning task when using both kinds of help types ($p - value = 0.04$).

We also use the TukeyHSD postoc analysis to discuss the differences in means of each help type. Here too, we only report the combinations that resulted in statistically significant differences. The differences in means between YY and NN is statistically significant ($p - value = 0.0005$). This indicates that using both the help video and the Interactive Human-aware Intervention for the planning tasks results in longer solutions compared to instances where no help is used. Furthermore, we find that the differences in means for YY and YN is also statistically significant ($p - value = 0.0007$). This indicates that given the human users watch the help video, using the Interactive Human-aware Intervention significantly increases the solution length, compared to instances where they do not use the Interactive Human-aware Intervention.

6.8 Discussion

We introduce four properties of the solution to the observer's planning problem $P_{observer}$ as hints: **the number of remaining moves**, **the next best move**, **the vehicles that must be moved** and **restart planning task**. The subjective preference ratings the human subjects gave for the helpfulness of the hints agree with the frequency the users requested the hint. For all there planning task types (C, E, M), the human subjects rated the "**Show the next best move**" as the highest in helpfulness. When considering the actual use of hints the hint "Show the next best move" hint was the most frequently requested hint for all three planning task types (C, E, M). This finding indicates that when human users are stuck during the Rush Hour planning task (i.e., getting close to moving the forbidden vehicle) they subjectively prefer information that tells directly them what to do next. In contrast, the "Number of remaining moves" and the "Vehicles that must be moved" are *heuristic information* of the planning problem $P_{observer}$. The heuristic information forces the human user to think deeply about the planning task instead of giving direct instructions. Our experiments show that the human users do not respond well to heuristic related information about the Rush Hour planning problem when they are offered as help.

We introduce four metrics to evaluate whether or not the Interactive Human-aware Intervention has been successful in helping the user find the solution to the planning task, while avoiding the forbidden vehicle as follows:

- The number of moves in the human user's solution
- The difference from the cost optimal solution (D_i)
- The landmark achievement ($\mathcal{L}_A$)
- The landmark regain ($\mathcal{L}_R$)

Although the 13 planning tasks we selected for this study produce optimal solutions having approximately the same number of moves (except for P8), the experiments show that the human users have difficulty getting closer to the optimal solution even with the use of hints. The longer the human users spend exploring the search space of the planning task, the higher the likelihood that he/she will also move the forbidden vehicle. When examining the cost optimal plans produced by an automated planner, we see that the landmarks are typically achieved in clusters. However, when examining the solutions produced by the human users, we see that they lose and regain the landmarks several times before eventually achieving them. The closer the human user's solution resembles the solution produced by the automated planner in terms of when the landmark clusters are achieved, the closer the number of moves will be to that of the optimal solution for that planning task. Therefore, information intended to guide the human user toward the solution to the Rush Hour planning task must be designed to help the user achieve the landmarks quicker. Furthermore, the guiding information needs to be presented to the user as direct instructions.

Let us now revisit the questions we intended to answer through our evaluation.

1. Does the Interactive Human-aware Intervention have an effect on the solution length?

Table 6.18: Summary of findings solving Rush Hour planning tasks with the use of Interactive Human-aware Intervention (condition) and without (control).

Dependent Variable	Factors	Result Between the Control and the Condition Groups
Number of Moves	Used the Interactive Human-aware Intervention (Yes/No) Planning task type (C, E, M)	Group means not statistically significant
Number of Moves	Used the Interactive Human-aware Intervention (Yes/No) Planning task ID (P1, P2, P4, P6, P7, P8, P9, P10)	Violates homogeneity of variance
D_i	Used the Interactive Human-aware Intervention (Yes/No) Planning task type (C, E, M)	Group means not statistically significant
D_i	Used the Interactive Human-aware Intervention (Yes/No) Planning task ID (P1, P2, P4, P6, P7, P8, P9, P10)	Violates homogeneity of variance
$\mathcal{L}_A$	Used the Interactive Human-aware Intervention (Yes/No) Planning task type (C, E, M)	Group means not statistically significant
$\mathcal{L}_A$	Used the Interactive Human-aware Intervention (Yes/No) Planning task ID (P1, P2, P4, P6, P7, P8, P9, P10)	Violates homogeneity of variance
$\mathcal{L}_R$	Used the Interactive Human-aware Intervention (Yes/No) Planning task type (C, E, M)	Group means not statistically significant
$\mathcal{L}_R$	Used the Interactive Human-aware Intervention (Yes/No) Planning task ID (P1, P2, P4, P6, P7, P8, P9, P10)	Violates homogeneity of variance

2. Does the Interactive Human-aware Intervention help move the user closer to the optimal solution?

3. Does seeing a help video affect the solution length?

Table 6.18 summarizes the analysis of the use of Interactive Human-aware Intervention during a Rush Hour planning task based on the four evaluation metrics. Our ideal expectation is that the Interactive Human-aware Intervention model reduces the number of moves, reduces $D - i$, reduces $\mathcal{L}_A$ and reduces $\mathcal{L}_R$. If the reductions between the control (did not use intervention) and the condition (used intervention) groups for our evaluation metrics are statistically significant, then it would have been possible to conclude that using the Interactive Human-aware Intervention helps move the user closer to the solution produced by an automated planner. Conversely, if we had observed significantly higher values for the evaluation metrics for the condition group compared to the control, then we would conclude that the Interactive Human-aware Intervention does not help move the user closer to the solution produced by a planner. In our analysis we we not able to find any statistical significance between the control and the condition groups for any of the evaluation metrics. Thus we conclude that considering the four evaluation metrics, the Interactive Human-aware Intervention neither improved or worsened the planning tasks.

When we resolve question 1 using the number of moves as the evaluation metric, we find that for some planning tasks types the Interactive Human-aware Intervention pushes the user closer to the optimal solution. However, the effect is not statistically significant considering the $\alpha = 0.05$ for the sample. When we resolve question 2 using the difference from the cost optimal solution, the landmark achievement and the landmark regain as evaluation metrics, we find that the differences from the cost optimal solution and the landmark achievement between those who used Interactive Human-aware Intervention and those who did not are not statistically significant between the three planning task types. We do not discuss the effect of the evaluation metrics for each of the eight planning tasks (P1, P2, P4, P6, P7, P8, P9, P10) in the control and the condition groups because the distributions violate the homogeneity of variances assumption for ANOVA.

When we resolve question 3 using the number of moves as the evaluation metric, we find that there is a statistically significant difference between the types of help (YY, NY, YN and NN) between the three planning task types: C, E and M. We use the data from the study where the subjects solved 13 planning tasks. Recall that the participants watch the video before they start the planning task and the Interactive Human-aware Intervention is used during the planning task. When we say that a subject *did not use the Interactive Human-aware Intervention* we mean that the subject did not see any hints being prompted by the Intervention agent during the planning task. We count all instances where the subject selected one of the hints prompted by the Intervention agent (including the Ignore hint option) as the subject using the Interactive Human-aware Intervention. The analysis shows that the different help types the users receive before and during the planning tasks significantly affects the number of moves in the solution. The posthoc analysis shows that for Type C planning tasks where the forbidden vehicle is positioned in the corner of the board, watching the video and also using the Interactive Human-aware Intervention results in a higher number of moves compared to instances where the human user do not use any type of help. Furthermore, using both kinds of help significantly increases the number of moves when compared to only using the video as help. This can be attributed to the fact that the hints force the user to think more deeply about the solution to the planning problem. Furthermore, the hint "Next Best Move" (most requested hint for type C planning tasks) moves the human user toward the cost optimal solution by only one step. This still leaves a lot of burden on the user to think more carefully about what the step after executing the hint should be, which they may not do willingly. The information given to the users through the "Next Best Move" hint may not be descriptive enough to move the user closer toward the cost optimal solution generated by an automated planner. Thus, it will be useful to explore the effect of revealing longer partial solutions to the remaining planning task to the user in order to help the user find the cost optimal plan faster. Of course, revealing the longest possible partial plan at once takes the cognitive load away from the user entirely. This is equivalent to the agent solving the planning task on the user's behalf (similar to a tutor revealing the answer to a math problem at once). In application domains where the human users wish to retain some autonomy while interacting with the AI system, this type of "giving away the solution" may not be preferable. On the other hand, revealing too little information, as we have encountered in this experiment, results in the user spending a long time searching for the solution. Therefore, when an automated planning agent interacts with the human user to assist the user complete a complex task, careful balance must be struck between the quality of the information revealed and the remaining planning task to determine: "*How much help is too much help*?". An important step toward addressing this question is to design metrics to evaluate the quality of information pertaining to the remaining planning task revealed to the user. In this research we have taken initial steps toward addressing this problem with the evaluation metrics derived from automated planning.

Posthoc analysis also revealed that there are significant differences in the solution lengths produced the three planning task types C, E and M. We find that solving type C planning tasks using both kinds of help (YY) take a significantly longer time compared to solving type E and M planning tasks using no help (NN). Although we can draw the conclusion that solving type C planning tasks with help takes longer compared to solving type E and M, when further examining the solutions produced by the human users, there are large variances in the number of moves within the planning tasks that fall into one category. For example, consider the planning task P2 and P8 in the same planning task type C, where the forbidden car is placed in the corner of the board. The minimum number of moves in a plan produced by a human user for task P2 is 67 and the maximum number of moves is 419. For task P8 the minimum number of moves is 9 and the maximum number of moves is 14. This observation allows us to further conclude that there are yet undiscovered factors about the Rush Hour planning tasks, besides the position of the forbidden vehicle, which contribute to the difficulty the human users experience in finding the solution. Some example factors that may determine the difficulty may be how blocked the forbidden vehicle is on the board, the number of vehicles blocking the path of the goal car. It is also likely that the difficulty may be determined by a combination of these factors.

We also individually compare the solution lengths of the thirteen planning tasks when the human users adopt different combinations of help types: YY, NY, YN and NN. However, due to the small number of participants in the groups, we do not report any statistical analysis results between the groups for the planning tasks.

We set a threshold for the number of steps for the Human-aware Intervention to start making predictions. The threshold is required to generate enough data for the classifier to predict intervention and minimize the false alarms early on in the planning task, which may cause the user to turn off the alerts. However, it was observed that many human users moved the forbidden vehicle within the window in which predictions were absent. This indicates that the human users need guidance early on in the planning task. It may be helpful to design intervention prediction models to reduce the size of this window.

There are external factors that affect the conclusion that Interactive Human-aware Intervention had a positive impact in helping the user complete the Rush Hour planning task. For example, if the participants are skilled at puzzle solving tasks in general or are familiar with the Rush Hour puzzle, they may not want or need the hints. In contrast, the participants who use the hints may find puzzle solving tasks difficult in general or may not be familiar with the Rush Hour puzzle. During our experiments we did not screen the users for their skill and familiarity for solving Rush Hour puzzles. This prevented us from factoring them in to the statistical analysis. Skill and familiarity may confound the results of the statistical analysis. Therefore, further studies may be required to fully understand their impact on the Interactive Human-aware Intervention and definitively conclude that the hints hurt rather than help the users.

Chapter 7

Concluding Remarks

In this dissertation we introduce the Intervention Problem and discuss three solutions addressing different properties of the problem. Intervention is important when an agent or a human user is executing tasks in an unfamiliar environment and unknown facts about the environment may cause the tasks to have unintended, perhaps dangerous consequences. It is also important that upon recognizing that intervention is needed, there is a system in place to guide the agent (or the human user) toward the goal, while avoiding the undesirable consequences. Thus, we propose intervention as a utility for online assistive agents and safety critical decision making for human users. The underlying assumption about the agent's (or the human user's) environment is that it can be modeled as a state transition system that consists of actions and states. Some states in the environment are undesirable and the agent (or the human user) does not have the ability to recognize them. An observer monitors what the agent is doing and wants to help the agent avoid the undesirable state.

7.1 The Intervention Recognition Problem

As illustrated in Figure 7.1, we model the Intervention Problem consisting of two sub problems. The first sub problem, known as **The Intervention Recognition Process**, the observer automatically detects that an undesirable state is developing. We present three intervention models each considering the properties: (1) actors in the environment, (2) goals hidden to the observer, (3) types of observations (4) noise in observations and (5) handling intervention recovery. In all three models, a key objective we want to address during the recognition process is ensuring the safety while allowing some freedom for the agent.

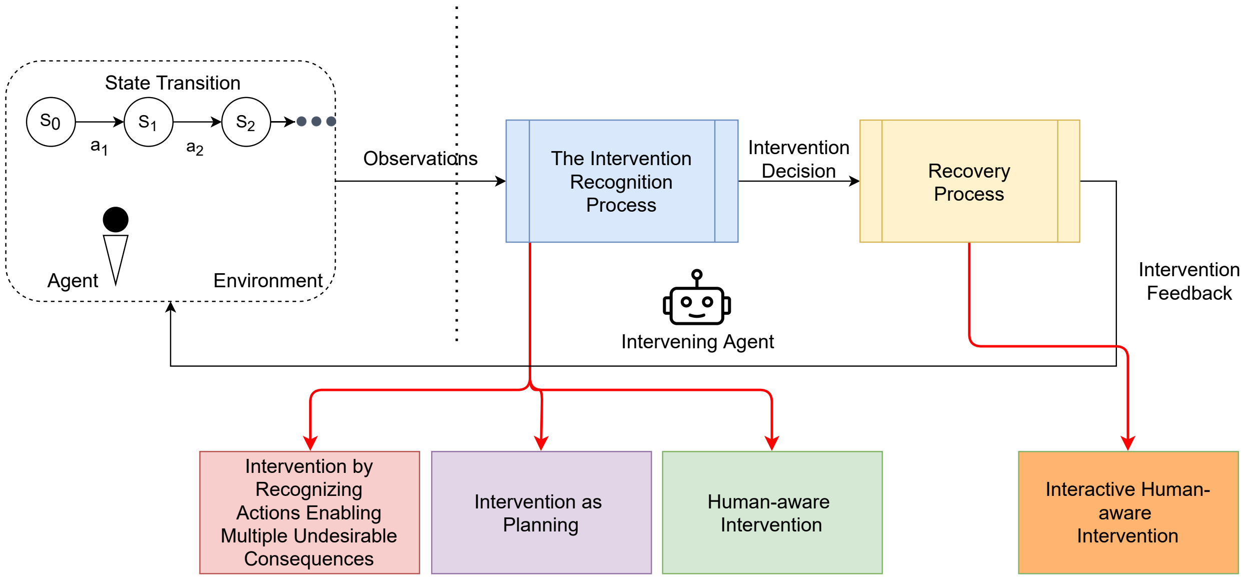


Figure 7.1: The Intervention Framework

7.1.1 Intervention by Recognizing Actions Enabling Multiple Undesirable Consequences

Using the cyber security domain as the motivation, we model an environment consisting of three agents: the user, the attacker and the observer. The observer wants to help the user reach a hidden desirable goal, while avoiding multiple undesirable states enabled by an attacker. Because the undesirable states are hidden to the user, he becomes an unwitting accomplice to security breaches. The observations that are used to make the intervention decision have missing and extraneous actions. The intervention recovery process is to simply block the recognized undesirable action by issuing an alert. Our approach views the intervention decision as a multi-factor decision problem modeled with three domain-independent metrics: **certainty**, **timeliness** and **desirability**. The observer projects the possible plans to reach the undesirable states using automated planning and traces back to find actions that are critical to the occurrence of the undesirable states. We simulate the trade-off between the safety and freedom of the agent (or the human user) when selecting actions requiring intervention by factoring in the domain-independent metrics with varying degrees of importance. Our experiments find that the certainty and desirability metrics deal well to the extraneous actions in the observation trace, while the timeliness metric is sensitive to the missing actions.

7.1.2 Intervention as Classical Planning

Using the benchmark planning domains, we model scenarios where the user the observer and an optional competitor attempt to achieve different goals in the same environment. A key difference in this form of intervention is that the observer is aware of the user's desirable goal and the undesirable state. Partial knowledge about the domain precludes the user from reaching his own goal and avoid facilitating the goal pursued by the competitor. In another scenario, the user may also unwittingly reach a hidden undesirable state on his own. The Intervention as Planning model allows the observer to intervene and guide the user towards his own goal while avoiding undesirable outcomes (i.e., the competitor's goal or hidden undesirable states) or frustration. The observer projects the possible plans to reach the undesirable and desirable states exactly and approximately using automated planning. To decide if intervention is necessary, the observer analyzes the plan suffixes leading to the user's desirable goal and the undesirable state. In one method, the observer uses the Intervention Graph, a data structure that represents the plan space to derive domain-independent metrics to analyze the suffixes of partially executed plans. In another method, the observer uses plan distance metrics to measure the differences between a projected plan hypothesis to the safe and the unsafe plans generated by an automated planner. In both cases, the observer uses the metrics to learn the differences between the safe and the unsafe suffixes (using off-the-shelf classification algorithms) and balance specific unsafe actions with those that are necessary for allowing the user some freedom. We compare the two intervention models to intervention by plan recognition for benchmark planning domains and show that our proposed approaches outperform three existing plan recognition algorithms.

7.1.3 Human-aware Intervention

In the two previous intervention models, we assume the user is an agent(s) using automated planners to generate the plans to achieve the desirable (and albeit by mistake, the undesirable states). When human users plan, especially for a cognitively taxing task, they do not have the ability to use heuristics to search for the best solutions. They often make mistakes and spend time exploring the search space of a planning problem, which in turn may cause them to accidentally reach undesirable states. To the Intervention Problem, what this means is that deciding to intervene by analyzing plan suffixes generated by an automated planner is not feasible when human users plan. Using the Rush Hour puzzle as the case study, we conduct a human subject experiment to analyze how human users solve a cognitively engaging puzzle as a planning task. We use the findings of this study to develop a Human-aware Intervention model combining automated planning and machine learning, where the observer must decide in real time whether to intervene for human user using a domain specific feature set more appropriate for human behavior. We show that the Human-aware Intervention model outperforms in recognizing undesirable outcomes in advance, compared to the existing plan recognition algorithms.

7.2 The Intervention Recovery Problem

The second sub problem, known as **The Recovery Process**, we go beyond the typical preventive measures to help the human user recover from intervention and interactively guide him toward safety using a new feedback technique called the **Interactive Human-aware Intervention**. In this planning task, the environment contains the human user and the observer who recognizes intervention using the Human-aware Intervention model. The observer is aware of the user's goal and also the undesirable states that are hidden from the user. We introduce a forbidden vehicle, which does not have to be moved to solve the Rush Hour planning task. Since the forbidden vehicle move is unnecessary, if the user moves that vehicle it indicates that the user has moved away from the solution to the planning task (the solution an automated planner would produce) and as a result needs intervention. We design an interactive feedback process where the observer, upon recognizing that the human user needs help, suggests information about the search space of the planning task as hints for the user. The hints are: **the number of remaining moves**, **the next best move**, **the vehicles that must be moved** and **restart the planning task**. We evaluate the effectiveness of the

Interactive Human-aware Intervention using the evaluation metrics that combine the cost optimal solutions produced by automated planners and also the planning landmarks:

1. The number of moves in the solution
2. The cost difference compared to the cost optimal solution
3. The latest time a fact landmark is eventually achieved
4. The number of times a fact landmark is lost and regained

When considering the number of moves as the evaluation metric, we observe that for some Rush Hour planning tasks, using the Interactive Human-aware Intervention reduces the solution length. However, this effect is not statistically significant compared to human users solving the same planning tasks without the Interactive Human-aware Intervention. For the evaluation metrics 2, 3, and 4, we observe a similar effect where for some Rush Hour planning tasks, using the Interactive Human-aware Intervention pushes the user closer to the optimal. However, here too, the effects are not statistically significant.

7.3 Future Work

Finally, we discuss possible extensions for our study of the Intervention problem.

7.3.1 Exploring Different Models of the Environment

The planning model is concerned with the Intervention Recognition sub problem, which requires some way to model how the agents (human or artificial) in the environment execute actions and how the actions modify the state. The intervention recognition models in our current implementations assume STRIPS planning models, where the world state is represented by a set of ground atomic literals and a plan is produced by chaining a sequence of ground actions, where each action has pre and post conditions. Another planning model that is useful, particularly in describing complex tasks is the Hierarchical Task Networks (HTN) (Erol, Hendler, Nau, & Tsuneto, 1995). The HTN planning model decomposes the events that take place in the environment into tasks and sub-tasks. The HTN planner receives a set of tasks to be performed and a plan is produced by repeatedly decomposing tasks into smaller and smaller sub-tasks until primitive tasks that can not be further decomposed are found. For intervention to be meaningful the scenario modeled in the planning domain must be complex, where one task may have several ways to be subverted (maliciously or by constraints in the environment). An example for a complex domain is Search and Rescue (SaR). In a SaR environment actors with specialized capabilities (e.g., police, victims, ambulance units, mission commanders) will collaborate to complete a rescue activity. All actors do not have full domain knowledge and as a result their actions may prevent other actors from reaching their own goals. SaR domain allows us to address situation-specific intervention where the intervention process need to account for different goals of agents collaborating to achieve a common higher-level goal.

7.3.2 Domain Abstraction Techniques to Explain Intervention

This issue focuses on the Intervention Recovery sub problem. The Interactive Human-aware Intervention model we have presented in this dissertation is different from the existing explanation types in the Explainable Artificial Intelligence Planning (XAIP), where the explainee can ask why-questions like “Why did you do action A?”, “Why didn’t you do something else that I would have done?” and “Why can’t you do that?”. For these questions, the XAIP literature suggests extracting contrastive or selective explanations from the planning domain to give an answer describing the cause and the effect relationship of the event in question (Miller, 2017). Our intention with the interactive Human-aware Intervention model is to guide the human user toward the solution, without giving away the solution directly. However, in complex planning domains the causal explanation model is also useful to help the user understand the intervention, by explaining what caused the intervention. The downside of complex planning domains, as required for intervention is that the plans used in the intervention decision process may be too difficult to understand for the human user. For example, in the cyber-security domain, many actions are possible, the human user only performs a limited set of actions. Even fewer number of actions trigger the security breaches. By simplifying the model the planner may find shorter plans, which in turn may result in simpler causal models for producing explanations. It is possible to explore the use of **macro-actions**: a domain abstraction technique to simplify the causal models and plans generated from complex domains (Newton et al., 2007).

7.3.3 Ensuring the Longevity of the Interactive Intervention Models

Intervention models designed for human user-agent collaborations must be robust to phenomena that have not been explicitly modeled. Even if the task is unfamiliar in the beginning, human users may learn over time, while interacting with the system The actor's goals may not be explicit, which requires the intervention agent to incorporate goal recognition as a prerequisite for the intervention model. Explanations for intervention must be catered toward human users having different expertise levels. Human users, and even artificial agents operating with partial information may make mistakes collaborating in a complex domain. Intervention can also be used to help the actors learn over time. It will be interesting to explore how the domain model itself can be learned from observations to account for human user's gaining experience and learning over time so that the longevity of the intervention model can be ensured. Maintaining portfolios of explainable models combining explainable black-box learning models and causal explainable models can be helpful in producing explanations catered toward end users with different domain levels of domain expertise.

THE END

References

2010 NCSA / Norton by Symantec online safety study. (2010). National Cyber Security Alliance and Nortan by Symantec and Zogby International.

Ajzen, I. (1991). The theory of planned behavior. *Organizational Behavior and Human Decision Processes*, *50*(2), 179–211.

Altmann, E. M., & Trafton, J. G. (2020). Memory for goals: An architectural perspective. In *Proceedings of the Twenty First Annual Conference of the Cognitive Science Society* (pp. 19–24). Psychology Press.

Amato, C., & Baisero, A. (2019). Active goal recognition. *ArXiv*, *abs/1909.11173*.

Ammann, P., Wijesekera, D., & Kaushik, S. (2002). Scalable, graph-based network vulnerability analysis. In *Proceedings of the Ninth ACM Conference on Computer and Communications Security* (pp. 217–224). Washington DC, USA.

Aytes, K., & Connolly, T. (2004). Computer Security and Risky Computing Practices: A Rational Choice Perspective. *Journal of Organizational and End User Computing*, *16*, 22–40.

Backstrom, C., & Klein, I. (1991). Parallel non-binary planning in polynomial time. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence* (Vol. 1, pp. 268–273). Sydney, New South Wales, Australia: Morgan Kaufmann Publishers Inc.

Banerjee, B., Kraemer, L., & Lyle, J. (2010). Multi-agent plan recognition: Formalization and algorithms. In *Proceedings of the Twenty Fourth AAAI Conference on Artificial Intelligence* (pp. 1059–1064). Atlanta, GA, USA: AAAI Press.

Bisson, F., Kabanza, F., Benaskeur, A., & Irandoust, H. (2011). Provoking opponents to facilitate the recognition of their intentions. In *Proceedings of the Twenty Fifth AAAI Conference on Artificial Intelligence* (pp. 1762–1763). San Francisco, CA, USA.

Blum, A. L., & Furst, M. L. (1997). Fast planning through planning graph analysis. *Artificial Intelligence*, *90*(1), 281–300.

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

Bonet, B., & Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, *129*(1), 5–33.

Borrajo, D., & Veloso, M. (2020). Domain-independent generation and classification of behavior traces. In *Working Notes of the ICAPS Workshop on Planning for Financial Services.* Nancy, France.

Briggs, G., & Scheutz, M. (2015). “Sorry, I Can’t Do That”: Developing mechanisms to appropriately reject directives in human-robot interactions. In *AAAI fall symposium on AI for human-robot interaction* (pp. 32–36). Arlington, VA, USA.

Bryce, D. (2014). Landmark-based plan distance measures for diverse planning. In *Proceedings of the Twenty Fourth International Conference on Automated Planning and Scheduling* (pp. 56–64). Portsmouth, NH, USA.

Byrne, Z. S., Dvorak, K. J., Peters, J. M., Ray, I., Howe, A., & Sanchez, D. (2016). From the user’s perspective: Perceptions of risk relative to benefit associated with using the Internet. *Computers in Human Behavior*, *59*, 456–468.

Chakraborti, T., Sreedharan, S., Grover, S., & Kambhampati, S. (2019, 22). Plan explanations as model reconciliation. In *The Fourteenth ACM/IEEE International Conference on Human-Robot Interaction* (pp. 258–266). Daegu, South Korea: IEEE Computer Society.

Chiasson, S., Forget, A., Biddle, R., & van Oorschot, P. C. (2008). Influencing users towards better passwords: Persuasive cued click-points. In *Proceedings of the 22nd British HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction* (Vol. 1, pp. 121–130). Liverpool, United Kingdom: BCS Learning and Development Ltd.

Claar, C. L. (2010). *The adoption of computer security: An analysis of home personal computer user behavior using the health belief model* (Unpublished doctoral dissertation). Utah State University.

Cohen, R., & Spencer, B. (1993). Specifying and updating plan libraries for plan recognition tasks. In *Proceedings of Ninth IEEE Conference on Artificial Intelligence for Applications* (Vol. 1, pp. 27–33). Orlando, FL, USA.

Coman, A., Gillespie, K., & Muñoz-Avila, H. (2015). Case-based local and global percept processing for rebel agents. In *Proceedings of the Twenty Third Annual International Conference on Case-based Reasoning, CEUR Workshop* (Vol. 1520, pp. 23–32).

Compeau, D. R., & Higgins, C. A. (1995). Computer self-efficacy: Development of a measure and initial test. *MIS Quarterly*, *19*(2), 189–211.

Computer and internet use in the united states. (2012). U.S. Census Bureau. Retrieved October, 2020, from `https://www.census.gov/data/tables/2012/demo/computer-internet/computer-use-2012.html`

Dannenhauer, D., Floyd, M. W., Magazzeni, D., & Aha, D. W. (2018). Explaining rebel behavior in goal reasoning agents. In *ICAPS Workshop on Explainable Planning* (pp. 12–18). Delft, The Netherlands.

Davinson, N., & Sillence, E. (2010). It won't happen to me: Promoting secure behaviour among internet users. *Computers in Human Behavior*, *26*(6), 1739–1747.

Debatin, B., Lovejoy, J. P., Horn, A.-K., & Hughes, B. N. (2009). Facebook and online privacy: Attitudes, behaviors, and unintended consequences. *Journal of Computer-mediated Communication*, *15*(1), 83–108.

Dupuis, M., Crossler, R., & Endicott-Popovsky, B. (2012). The information security behavior of home users: Exploring a user's risk tolerance and past experiences in the context of backing up information. In *The Dewald Roode Information Security Workshop* (p. 33). Provo, UT, USA.

E-Martín, Y., R-Moreno, M. D., & Smith, D. E. (2015). A fast goal recognition technique based on interaction estimates. In *Proceedings of the Twenty Fourth International Conference on Artificial Intelligence* (pp. 761–768). Buenos Aires, Argentina: AAAI Press.

Erol, K., Hendler, J. A., Nau, D. S., & Tsuneto, R. (1995). A critical look at critics in htn planning. In *Proceedings of the International Joint Conference on Artificial Intelligence.* Montreal, Quebec, Canada.

Fernau, H., Hagerup, T., Nishimura, N., Ragde, P., & Reinhardt, K. (2003). On the parameterized complexity of the generalized rush hour puzzle. In *Proceedings of the Fifteenth Canadian Conference on Computational Geometry* (pp. 6–9). Halifax, Nova Scotia, Canada.

Fikes, R. E., & Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*(3), 189–208.

Flake, G. W., & Baum, E. B. (2002). Rush hour is PSAPCE-complete, or "Why you should generously tip parking lot attendants". *Theoritical Computer Science*, *270*(1-2), 895–911.

Fox, M., Long, D., & Magazzeni, D. (2017). Explainable planning. In *Proceedings of the International Joint Conference on Artificial Intelligence Workshop on Explainable AI.* Melbourne, Australia.

Freedman, R. G., & Zilberstein, S. (2017). Integration of planning with recognition for responsive interaction using classical planners. In *Proceedings of the Thirty First AAAI Conference on Artificial Intelligence* (pp. 4581–4588). AAAI Press.

Furnell, S., Bryant, P., & Phippen, A. (2007). Assessing the security perceptions of personal Internet users. *Computers & Security*, *26*(5), 410–417.

Gal, Y., Reddy, S., Shieber, S. M., Rubin, A., & Grosz, B. J. (2012). Plan recognition in exploratory domains. *Artificial Intelligence*, *176*(1), 2270–2290.

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

Geib, C. W., & Goldman, R. P. (2005). Partial observability and probabilistic plan/goal recognition. In *Proceedings of the International Workshop on Modeling Other Agents From Observations* (Vol. 8, pp. 1–6).

Ghallb, M., Howe, A., Knoblock, C., McDermott, D., Ram, A., Veloso, M., ... Wilkins, D. (1998). *PDDL - The Planning Definition Language* (Tech. Rep.). CT, USA: Yale Center for Computational Vision and Control.

Good, N., Dhamija, R., Grossklags, J., Thaw, D., Aronowitz, S., Mulligan, D., & Konstan, J. (2005). Stopping spyware at the gate: a user study of privacy, notice and spyware. In *Proceedings of the 2005 symposium on Usable privacy and security* (pp. 43–52).

Govani, T., & Pashley, H. (2005). *Student awareness of the privacy implications when using Facebook.* Unpublished paper presented at the "Privacy Poster Fair" at the Carnegie Mellon University School of Library and Information Science. Retrieved September 2020, from `http://lorrie.cranor.org/courses/fa05/tubzhlp.pdf`

Gregg-Smith, A., & Mayol-Cuevas, W. W. (2015). The design and evaluation of a cooperative handheld robot. In *Proceedings of the IEEE International Conference on Robotics and Automation* (pp. 1968–1975).

Gupta, N., & Nau, D. S. (1992). On the complexity of blocks-world planning. *Journal of Artificial Intelligence*, *56*(2), 223–254.

Hadfield-Menell, D., Russell, S. J., Abbeel, P., & Dragan, A. (2016). Cooperative inverse reinforcement learning. *Advances in Neural Information Processing Systems*, *29*, 3909–3917.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA Data Mining Software: An Update. *SIGKDD Exploration Newsletter*, *11*(1), 10–18.

Helmert, M. (2006). The fast downward planning system. *Journal of Artificial Intelligence Research*, *26*(1), 191–246.

Helmert, M., & Domshlak, C. (2009). Landmarks, critical paths and abstractions: What's the difference anyway? *Proceedings of the International Conference on Automated Planning and Scheduling*, *19*(1), 162–169.

Hiatt, L. M., Harrison, A. M., & Trafton, J. G. (2011). Accommodating human variability in human-robot teams through theory of mind. In *Proceedings of the Twenty Second International Joint Conference on Artificial Intelligence* (Vol. 3, pp. 2066–2071). Barcelona, Spain: AAAI Press.

Hoffmann, J. (2003). The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*, *20*(1), 291–341.

Hoffmann, J., Porteous, J., & Sebastia, L. (2004). Ordered landmarks in planning. *Journal of Artificial Intelligence Research*, *22*, 215–278.

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

Howe, A. E., Ray, I., Roberts, M., Urbanska, M., & Byrne, Z. (2012). The psychology of security for the home computer user. In *Proceedings of the IEEE Symposium on Security and Privacy* (pp. 209–223).

How to recognize and avoid phishing scams. (2019). Federal Trades Commission. Retrieved October, 2020, from `https://www.consumer.ftc.gov/articles/how-recognize-and-avoid-phishing-scams#recognize`

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*(1), 99–134.

Kaminka, G. A., Pynadath, D. V., & Tambe, M. (2002). Monitoring teams by overhearing: A multi-agent plan-recognition approach. *Journal of Artificial Intelligence Research*, *17*(1), 83–135.

Kautz, H. A., & Allen, J. F. (1986). Generalized plan recognition. In *Proceedings of Fifth National Conference on Artificial Intelligence* (pp. 32–37). Philadelphia, PA, USA.

Keren, S., Gal, A., & Karpas, E. (2014). Goal recognition design. In *Proceedings of the Twenty Fourth International Conference on Automated Planning and Scheduling* (pp. 154–162). Portsmouth, NH, USA.

Langley, P., Meadows, B., Sridharan, M., & Choi, D. (2017). Explainable agency for intelligent autonomous systems. In *Proceedings of the Twenty Ninth AAAI Conference on Innovative Applications Conference* (pp. 4762–4763). Austin, TX, USA.

Lesh, N., & Etzioni, O. (1996). Scaling up goal recognition. In *Proceedings of the Fifth International Conference on Knowledge Representation and Reasoning* (pp. 178–189). Cambridge, MA, USA.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Proceedings of the Thirty First International Conference on Neural Information Processing Systems* (pp. 4768–4777). Red Hook, NY, USA: Curran Associates Inc.

Mannan, M., & Oorschot, P. V. (2008). Security and usability: the gap in real-world online banking. In *Working Notes of the New Security Paradigms Workshop.* New Hampshire, USA.

Miller, T. (2017). Explanation in artificial intelligence: Insights from the social sciences. *arXiv preprint*, *1706.07269*.

Milne, G. R., Labrecque, L. I., & Cromer, C. (2009). Toward an understanding of the online consumer's risky behavior and protection practices. *Journal of Consumer Affairs*, *43*(3), 449–473.

Mirsky, R., Stern, R., Gal, K., & Kalech, M. (2018). Sequential plan recognition: An iterative approach to disambiguating between hypotheses. *Artificial Intelligence*, *260*, 51–73.

Nau, D., Ghallab, M., & Traverso, P. (2004). *Automated Planning: Theory & Practice*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Newton, M., Levine, J., Fox, M., Long, D., Boddy, M., Fox, M., & Thiebaux, S. (2007). Learning macro-actions for arbitrary planners and domains. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling* (pp. 256–263).

Ng, B. Y., & Xu, Y. (2007). Studying users' computer security behavior using the health belief model. *Decision Support Systems*, *46*(4), 815–825.

Nguyen, T. A., Do, M., Gerevini, A. E., Serina, I., Srivastava, B., & Kambhampati, S. (2012). Generating diverse plans to handle unknown and partially known user preferences. *Artificial Intelligence*, *190*, 1–31.

of Southern California Annenberg School Center for the Digital Future, U. (2018). *The world internet project international report (ninth edition).* Retrieved October 2020, from `https://www.digitalcenter.org/wp-content/uploads/2019/01/`

World-Internet-Project-report-2018.pdf

Parliament, E., & Council. (2016). Regulation on theprotection of natural persons with regard to the processing of personal data and on the free movement of such data, andrepealing directive 95/46/ec (general data protection regulation). *Official Journal of the European Union*, *L119*, 1–88.

Pereira, R. F., Oren, N., & Meneguzzi, F. (2017). Landmark-based heuristics for goal recognition. In *Proceedings of the Thirty First Association for the Advancement of Artificial Intelligence* (pp. 3622–3628). San Francisco, CA, USA.

Pozanco, A., Yolanda, E., Fernández, S., & Borrajo, D. (2018). Counterplanning using goal recognition and landmarks. In *Proceedings of the Twenty Seventh International Joint Conference on Artificial Intelligence* (pp. 4808–4814). Stockholm, Sweden.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Ramírez, M., & Geffner, H. (2009). Plan recognition as planning. In *Proceedings of the Twenty First International Joint Conference on Artifical Intelligence* (pp. 1778–1783). Pasadena, CA, USA.

Ramírez, M., & Geffner, H. (2010). Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence* (pp. 1121–1126). Atlanta, GA, USA.

Ramírez, M., & 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* (pp. 2009–2014). Barcelona, Spain: AAAI Press.

Ratwani, R. M., McCurry, J. M., & Trafton, J. G. (2008). Predicting postcompletion errors using eye movements. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 539–542). Association for Computing Machinery.

Riabov, A., Sohrabi, S., & Udrea, O. (2014). New algorithms for the Top-K planning problem. In *Proceedings of the the Scheduling and Planning Applications woRKshop (SPARK) at the*

Twenty Fourth International Conference on Automated Planning and Scheduling (pp. 10–16). Portsmouth, NH, USA.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the Twenty Second ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144). New York, NY, USA: ACM.

Rich, C., Sidner, C. L., & Lesh, N. (2001). Collagen: Applying collaborative discourse theory to human-computer interaction. *AI magazine*, *22*(4), 15–25.

Richter, S., & Westphal, M. (2010). The LAMA planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research*, *39*, 127–177.

Roberts, M., Howe, A., & Ray, I. (2014). Evaluating diversity for classical planning. In *Proceedings of the International Conference on Automated Planning and Scheduling* (pp. 253–261). Portsmouth, NH, USA: AAAI Press.

Rogers, R. W., & Prentice-Dunn, S. (1997). Protection motivation theory. *Handbook of Health Behavior Research 1: Personal and Social Determinants*, 113–132.

Roschke, S., Cheng, F., & Meinel, C. (2011). A new alert correlation algorithm based on attack graph. In Á. Herrero & E. Corchado (Eds.), *Computational Intelligence in Security for Information Systems* (pp. 58–67). Berlin, Heidelberg: Springer Berlin Heidelberg.

Rosenstock, I. M., Strecher, V. J., & Becker, M. H. (1988). Social learning theory and the health belief model. *Health Education Quarterly*, *15*(2), 175–183.

Ryan, C. (2017). *Computer and Internet Use in the United States: 2016* (American Community Survey Reports, ACS-39). Washington, DC: U.S. Census Bureau.

Saisubramanian, S., Kamar, E., & Zilberstein, S. (2020). A multi-objective approach to mitigate negative side effects. In C. Bessiere (Ed.), *Proceedings of the Twenty Ninth International Joint Conference on Artificial Intelligence* (pp. 354–361). Yokohama, Japan: International Joint Conferences on Artificial Intelligence Organization.

Sasse, M. A., Brostoff, S., & Weirich, D. (2001). Transforming the “Weakest Link” – A Human/computer Interaction Approach to Usable and Effective Security. *BT Technology Journal*, *19*(3), 122–131.

Schmidt, C. F., Sridharan, N., & Goodson, J. L. (1978b). The plan recognition problem: An intersection of Psychology and Artificial Intelligence. *Artificial Intelligence*, *11*(1-2), 45–83.

Schmidt, C. F., Sridharan, N. S., & Goodson, J. L. (1978a). The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence*, *11*(1–2), 45–83.

Sheyner, O., Haines, J., Jha, S., Lippmann, R., & Wing, J. (2002). Automated generation and analysis of attack graphs. In *Proceedings of the IEEE Symposium on Security and Privacy* (pp. 273–284). Washington, DC, USA: IEEE Computer Society.

Shvo, M., Klassen, T. Q., Sohrabi, S., & McIlraith, S. A. (2020). Epistemic plan recognition. In *Proceedings of the Nineteenth International Conference on Autonomous Agents and Multi-Agent Systems* (pp. 1251–1259). Aukland, New Zealand.

Shvo, M., & McIlraith, S. A. (2020). Active goal recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, pp. 9957–9966). New York, NY, USA.

Shvo, M., Sohrabi, S., & McIlraith, S. (2018). An AI Planning-Based Approach to the Multi-Agent Plan Recognition Problem (Preliminary Report). In *Advances in ArtificialIntelligence: Thirty First Canadian Conference on Artificial Intelligence* (pp. 253–258). Toronto, ON, Canada.

Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2020). Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society.* Retrieved from https://arxiv.org/pdf/1911.02508.pdf

Sohrabi, S., Baier, J., & McIlraith, S. (2011). Preferred explanations: Theory and generation via planning. In *The Twenty Fifth AAAI Conference on Artificial Intelligence.* San Francisco, CA, USA.

Sohrabi, S., Riabov, A., & Udrea, O. (2016). Plan recognition as planning revisited. In *Proceedings of the Twenty Fifth International Joint Conference on Artificial Intelligence* (pp. 3258–3264). New York, NY, USA: AAAI Press.

Sohrabi, S., Riabov, A. V., Udrea, O., & Hassanzadeh, O. (2016). Finding diverse high-quality plans for hypothesis generation. In *Proceedings of the Twenty Second European Conference on Artificial Intelligence* (pp. 1581–1582). The Hague, Holland.

Sotirakopoulos, A., Hawkey, K., & Beznosov, K. (2011). On the challenges in usable security lab studies: Lessons learned from replicating a study on SSL warnings. In *Proceedings of the Seventh Symposium on Usable Privacy and Security.* Pittsburgh, Pennsylvania: Association for Computing Machinery.

Sun, J., & Yin, M. (2007). Recognizing the agent's goals incrementally: planning graph as a basis. *Frontiers of Computer Science in China*, *1*(1), 26–36.

Talamadupula, K., Briggs, G., Chakraborti, T., Scheutz, M., & Kambhampati, S. (2014). Coordination in human-robot teams using mental modeling and plan recognition. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2957–2962). Chicago, IL, USA.

Urbanska, M., Roberts, M., Ray, I., Howe, A., & Byrne, Z. (2013). Accepting the inevitable: Factoring the user into home computer security. In *Proceedings of the Third ACM Conference on Data and Application Security and Privacy.* San Antonio, TX, USA.

Vattam, S. S., & Aha, D. W. (2015). Case-based plan recognition under imperfect observability. In *Proceedings of the Twenty Third International Conference on Case Based Reasoning* (pp. 381–395). Cham, Germany: Springer.

Vaughan-Nichols, S. J. (2020). *Most popular operating systems of 2020: The more things change...* Retrieved October 2020, from `https://www.zdnet.com/article/whats-2020s-most-popular-operating-systems/`

Vered, M., & Kaminka, G. A. (2017). Heuristic online goal recognition in continuous domains. In *Proceedings of the Twenty Sixth International Joint Conference on Artificial Intelligence* (pp. 4447–4454). Melbourne, Australia.

Vered, M., Pereira, R. F., Magnaguagno, M., Meneguzzi, F., & Kaminka, G. A. (2018). Online goal recognition as reasoning over landmarks. In *Working Notes of the The AAAI 2018 Workshop on Plan, Activity, and Intent Recognition.* New Orleans, LA, USA.

Virvou, M., & Kabassi, K. (2002a). IFM: An intelligent graphical user interface offering advice. In *Proceedings of the Second Hellenic Conference of AI* (pp. 155–164). Thessaloniki, Greece.

Virvou, M., & Kabassi, K. (2002b). Reasoning about users' actions in a graphical user interface. *Human-computer Interaction*, *17*, 369–398.

Warkentin, M., Johnston, A. C., Shropshire, J., & Barnett, W. D. (2016). Continuance of protective security behavior: A longitudinal study. *Decision Support Systems*, *92*, 25–35. (A Comprehensive Perspective on Information Systems Security - Technical Advances and Behavioral Issues)

Weber, E. U., Blais, A.-R., & Betz, N. E. (2002). A domain-specific risk-attitude scale: measuring risk perceptions and risk behaviors. *Journal of Behavioral Decision Making*, *15*(4), 263–290.

Weinstein, N. D., & Sandman, P. M. (2002). The precaution adoption process model and its application. *Emerging Theories in Health Promotion Practice and Research*, 16–39.

Wilkins, D. E., Lee, T. J., & Berry, P. (2003). Interactive execution monitoring of agent teams. *Journal of Artificial Intelligence Research*, *18*, 217-261.

Witte, K. (1992). Putting the fear back into fear appeals: The extended parallel process model. *Communications Monographs*, *59*(4), 329–349.

Yadav, A., Chan, H., Xin Jiang, A., Xu, H., Rice, E., & Tambe, M. (2016). Using social networks to aid homeless shelters: Dynamic influence maximization under uncertainty. In *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems* (pp. 740–748). Singapore, Singapore: International Foundation for Autonomous Agents and Multiagent Systems.

Yin, J., Chai, X., & Yang, Q. (2004). High-level goal recognition in a wireless LAN. In *Proceedings of the National Conference on Artificial Intelligence* (pp. 578–584). San Jose, CA, USA.

Zhang, S., Durfee, E. H., & Singh, S. (2018). Minimax-regret querying on side effects for safe optimality in factored markov decision processes. In *Proceedings of the Twenty Seventh International Joint Conference on Artificial Intelligence* (pp. 4867–4873). AAAI Press.

Appendix A

Cyber-security: Pre-survey

Dear Participant,

Thank you for participating in our user computer usage study. Please complete the following questionnaire as accurately as possible before completing the experiment tasks. The purpose of this questionnaire is to gather information about your computer usage patterns. The information you provide will be kept anonymous.

Please answer ALL questions.

1. State your age in years

   o 20 – 25
   o 26 – 30
   o 31 – 35
   o 36 – 40
   o 41 and above
   o I do not want to give that information

2. State your gender

   o Male
   o Female
   o I do not want to give that information

3. State your education level

   o Not currently in college - no bachelors or associates degree completed
   o Currently in college and working towards an associates degree
   o Currently in college and working towards a bachelors degree
   o Not currently in college - have completed bachelors degree within the last 5 years
   o Currently in college and working towards a graduate degree
   o Not currently in college - have a completed graduate degree
   o Other ____________________
   o I do not want to give that information

4. How long have you been using computers?

   o 1 – 2 years
   o 3 – 4 years

- o 5 – 6 years
- o more than 6 years

5. How many hours per day do you use the computer?

    - o less than 2 hours
    - o 2 – 5 hours
    - o 5 – 8 hours
    - o more than 8 hours

6. Which of the following computing devices do you own? (Select all that apply)

    - o Desktop PC
    - o Laptop
    - o Smart Phone
    - o Tablet PC
    - o E-reader
    - o Other ____________________

7. What software applications do you use regularly? (Select all that apply)

    - o Web browser (Internet Explorer, Firefox, Chrome)
    - o Office application suite (Word processors, spreadsheets etc.)
    - o Media players (windows media player, QuickTime, VLC media player etc.)
    - o Adobe Acrobat Reader (for PDF files)
    - o Computer games
    - o Design and image processing applications (Adobe Photoshop, Flash, GIMP etc.)
    - o Software development tools (Eclipse, Netbeans, IntelliJ IDEA)
    - o Other ____________________

8. For what purposes do you often use the computer? (Select all that apply)

    - o work/business
    - o Education
    - o Gathering information from the Internet (online news, weather, sports etc.)
    - o Communicating with others (instant messaging, email, Facebook, Twitter etc)
    - o Preparing documents, spreadsheets, presentations
    - o Playing computer games
    - o Financial activities (e.g. banking, online shopping, budgeting etc.)
    - o Programming and other software design and development tasks
    - o Other ____________________

9. Which of the following tasks related to computer usage do you find most challenging? (Select all that apply)

- o Finding application software that matches my requirements
- o Installing, configuring and getting a software application ready to be used
- o Finding and using help manuals
- o Identifying actions that may harm the computer
- o Taking steps to ensure the safety of the computer
- o Other ____________________

10. Have you had any formal training to use the computer?

    - o I have taken computer programming courses (in college/university, online)
    - o I have taken courses in using computer applications (in college/university, online)
    - o I have followed online tutorials to learn how to use the computer
    - o I have taken part in face-to-face seminars/tutorial classes on using the computer
    - o Other ____________________

11. Where do you go when you need help in how to use the computer? (Select all that apply)

    - o Internet
    - o Relatives
    - o Friends
    - o Local stores (e.g., Apple Genius Bar)
    - o Other ____________________

12. Have you installed any anti-virus software in your personal computer?

    - o Yes
    - o No

13. Which of the following commonly used software have you installed in your personal computer? (Select all that apply)

    - o Google Chrome
    - o Mozilla Firefox
    - o Microsoft Office Package (Microsoft Word, Microsoft Excel, PowerPoint etc.)
    - o Opensource Office Package (LibreOffice, OpenOffice)
    - o Apple QuickTime media player
    - o VLC Media Player
    - o Adobe Flash Player
    - o Adobe Acrobat Reader
    - o Skype
    - o Third-party email clients (Thunderbird, Eudora, Outlook Express)
    - o Other ____________________

Appendix B

Cyber-security: Post-survey

1. What was the full name you used for this study?
2. How difficult was it for you complete the Install and configure Antivirus software task?

   Very Hard Hard Neither easy nor hard Easy Very easy
3. I feel confident in my ability to configure and use an antivirus software.

   Strongly Disagree Disagree Neutral Agree Strongly agree
4. I feel confident in my ability to select an antivirus software that matches my security requirements.

   Strongly Disagree Disagree Neutral Agree Strongly agree
5. I feel confident in my ability to identify legitimate antivirus software.

   Strongly Disagree Disagree Neutral Agree Strongly agree
6. I feel confident in my ability to identify and remove suspicious files on my computer using an antivirus software.

   Strongly Disagree Disagree Neutral Agree Strongly agree
7. I feel confident in my ability to identify and remove suspicious files on my computer using an antivirus software without help.

   Strongly Disagree Disagree Neutral Agree Strongly agree
8. Before choosing a source to download software, I first check if it is hosted by a trustworthy provider as it may expose me to a security threat.

   Strongly Disagree Disagree Neutral Agree Strongly agree
9. I choose an antivirus software, which allows me to customize its features to match my security requirements.

   Strongly Disagree Disagree Neutral Agree Strongly agree
10. I always use an antivirus software, configured to my preferred settings in my computer system.

Strongly Disagree Disagree Neutral Agree Strongly agree

11. Using an antivirus software to identify and remove suspicious files in my computer is important to me.

    Strongly Disagree Disagree Neutral Agree Strongly agree

12. How difficult was it for you complete the Respond to direct messages from twitter account task?

    Very Hard Hard Neither easy nor hard Easy Very easy

13. I feel confident in my ability to use the direct messaging feature in Twitter.

    Strongly Disagree Disagree Neutral Agree Strongly agree

14. I feel confident in my ability to identify suspicious messages on Twitter.

    Strongly Disagree Disagree Neutral Agree Strongly agree

15. I feel confident in my ability to identify suspicious messages on Twitter without help.

    Strongly Disagree Disagree Neutral Agree Strongly agree

16. I feel confident in my ability to identify suspicious messages on Twitter even if it is the first time I had seen one.

    Strongly Disagree Disagree Neutral Agree Strongly agree

17. Before following a link on Twitter, I first check if it was sent by an unknown sender or contains suspicious information.

    Strongly Disagree Disagree Neutral Agree Strongly agree

18. I practice caution when I follow links sent to me on messages on Twitter as they may expose me to a security threat.

    Strongly Disagree Disagree Neutral Agree Strongly agree

19. I do not follow links sent to me on Twitter messages, if the content looks suspicious.

    Strongly Disagree Disagree Neutral Agree Strongly agree

20. When I see a message warning me about the security of a web site (e.g., expired certificate, http:// instead of https://) I am about to visit, I do not visit that web site.

    Strongly Disagree Disagree Neutral Agree Strongly agree

21. I practice caution when responding to email from senders I am not familiar with.

    Strongly Disagree Disagree Neutral Agree Strongly agree

22. I practice caution downloading attachments in emails from senders I am not familiar with.

    Strongly Disagree Disagree Neutral Agree Strongly agree

23. When prompted to change default passwords I always change the password.

    Strongly Disagree Disagree Neutral Agree Strongly agree

24. When I am changing passwords, I am confident in my ability to choose a strong password.

    Strongly Disagree Disagree Neutral Agree Strongly agree

25. Are you capable of recognizing secure web pages from non-secure web pages?

    Yes No I do not know

26. Was Twitter.com page you saw during the experiment a secure web page?

    Yes No I do not know

27. When you are asked to submit private information to websites, are you concerned about the safety of information you give?

    Yes No I do not know

28. When you submit private information (passwords, email addresses) to websites, do you verify that the site is secure and your information is safe?

    Yes No I do not know

29. During the experiment did you download the antivirus software from a secure web page?

    Yes No I do not know

30. The descriptions given for two anti-virus programs (Security Shield and PCDoctor) were helpful in choosing what software to install.

    Strongly Disagree Disagree Neutral Agree Strongly agree

31. How do you recognize a legitimate web page hosted by a trusted party from a illegitimate, unsafe website? (Select all that apply)

    o professional appearance (e.g, less flashy, formal writing etc.)
    o contact information (email, address, phone numbers) specified on the website
    o URL starting with https://
    o Content, and photos on the website look original and related to the host's products

o Not displaying advertisements that belong to third parties
o website is well known and popular
o I do not know

32. How do you recognize harmful email (spam/phishing)? (Select all that apply)

    o Email address is from a unrecognizable/unheard of domain and not relevant to the person or company represented in the email
    o contact information (email, address, phone numbers) specified on the website
    o Links on the email are not legitimate and not relevant to the person or company represented in the email
    o Incorrect grammar/spelling
    o Email greeting is not personalized (e.g., Dear Customer instead of Dear (Your Name))
    o Email message contains attachments that seem irrelevant to email message
    o Email messages with threats (e.g., discontinuing a service if not responded, changing passwords and other private information without your acknowledgment etc.)
    o I do not know

33. When installing a software do you usually read the end user license agreement?

    Yes   No

34. If you answered NO, why did you decide not to read the license agreement? (Select the choice that seems MOST important)

    o It is too long to read
    o It is worded in technical terms I do not understand
    o I do not ever read license agreements and no harm has happened to me
    o I have no choice but to click on accept, if I want to use the software
    o I do not think information on the license agreement is relevant to me

35. Did you read the end user license agreement when downloading the anti-virus software during the experiment task?

    Yes   No

Appendix C

Rush Hour Intervention Design: Post-survey

Dear Participant,

Thank you for participating in our study. Please complete the following questionnaire as accurately as possible. The purpose of this questionnaire is to gather information about your interests in logic puzzle solving (e.g., Rush Hour, Sudoku, sliding puzzles). The information you provide will be kept anonymous.

Please answer ALL questions.

1. State your age in years

   o less than 20
   o 20 – 25
   o 26 – 30
   o 31 – 35
   o 36 – 40
   o 41 and above
   o I do not want to give that information

2. State your gender

   o Male
   o Female
   o I do not want to give that information

3. Do you like solving logic puzzles in general (e.g., Sudoku, Sliding block puzzles)?

   Strongly Disagree   Disagree   Neutral   Agree   Strongly agree

4. Did you like the Rush Hour puzzle solving task?

   Strongly Disagree   Disagree   Neutral   Agree   Strongly agree

5. How many hours per day do you use the computer?

   o less than 2 hours
   o 2 – 5 hours
   o 5 – 8 hours

o more than 8 hours

6. What about solving logic puzzles that interest you the most? (Select all that apply)

   o Fun and relaxing
   o Sharpens my critical thinking skills
   o Challenge my friends and family to beat my time/score
   o Other ____________________

7. How often do you solve logic puzzles?

   o More than once a day
   o Once a day
   o Once every two to three days
   o Once a week
   o Once every two to three weeks
   o Once a month
   o Other ____________________

8. If the puzzle you are solving appear to be difficult, what do you most often do?

   o Give up
   o Keep trying until I solve it
   o Ask for help (help guides, online forums etc.)
   o Other ____________________

9. Imagine you are solving an all new puzzle that you have not seen before. If you were to get stuck while solving the puzzle, which of these options would you most like to be available?

   o A suggestion/tip to help me get past the problem I am in now
   o A timely warning that lets me know my current strategy is faulty
   o A timely warning plus an explanation on how to avoid similar problems in the future
   o I do not want outside help. I want to solve a puzzle on my own
   o Other ____________________

10. What do you play logic puzzles on? (Select all that apply)

   o Mobile
   o Gaming consoles (e.g., XBox)
   o Personal computer/laptop
   o Puzzle books, magazines, news papers
   o Other ____________________

Appendix D

Rush Hour Intervention Hints: Post-survey

Dear Participant,

Thank you for participating in our study. Please complete the following questionnaire as accurately as possible. The purpose of this questionnaire is to gather information about your general impression on helpful hints that were produced during the Rush Hour puzzle solving task. The information you provide will be kept anonymous.

Please answer ALL questions.

1. State your age in years

   o less than 20
   o 20 – 25
   o 26 – 30
   o 31 – 35
   o 36 – 40
   o 41 and above
   o I do not want to give that information

2. What is your education background?

   o Arts/Humanities
   o Business
   o Computer Science
   o Engineering
   o Other ____________________

3. Have you solved a Rush Hour puzzle before?

   o Yes
   o No

4. Did you see hint notifications during the puzzle solving task?

   o Yes
   o No

5. The hints helped me avoid moving the forbidden vehicle while solving the puzzle BEFORE I knew what the forbidden vehicle was.

Strongly Disagree Disagree Neutral Agree Strongly agree

6. Rate the four hints based on how helpful they were for you to avoid moving the forbidden vehicle
   - o Hint 1: See the minimum remaining number of moves

     (low) 1 2 3 4 5 (high)
   - o Hint 2: See the next best move

     (low) 1 2 3 4 5 (high)
   - o Hint 3: See the vehicles that must be moved

     (low) 1 2 3 4 5 (high)
   - o Hint 4: Restart puzzle

     (low) 1 2 3 4 5 (high)

Appendix E

Rush Hour Intervention Human Subject Experiment

We discuss details about the Rush Hour human subject study and the findings. The following sections present the evidence on which we based the design choices for the Rush Hour planning task. The sections are cross-referenced throughout the main content.

E.1 Rush Hour Human Subject Study: Pilot Experiment

Before the study was actually administered to our recruited subjects, a pilot study was conducted with nine graduate students to assess whether they would be able to solve the puzzles within a reasonable amount of time. The pilot study participants solved their assigned puzzles within 5-10 minutes. The pilot study participants were also interviewed informally to get their perception of the puzzle difficulty. The participants commented that the puzzles were "*challenging*" and "*forced me to think*". The same puzzles used in the pilot were used in the actual study.

E.2 Experimental Design

For the actual study, we recruited subjects from a university student population. The participants were not compensated for their time. After obtaining informed consent, the participants were directed to the Web URL, which hosted the Rush Hour Web Simulator. Each participant was assigned to solve one Rush Hour puzzle randomly selected from a set of ten. We did not place any time restriction for the puzzle solving task. Participants also had the option to use an online tutorial. All ten puzzles contained a forbidden vehicle. The participants were informed that one of the vehicles on the game board is forbidden and the puzzle must be solved without moving the forbidden vehicle. However, in this phase we did not give any visual cues (error messages, blocks, intervention) to the user in case they happen to move the forbidden vehicle during game play. Once the puzzle solving task was completed, the participants were asked to complete a short demographic

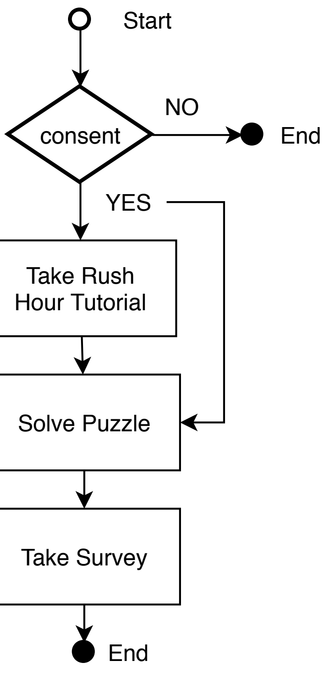


Figure E.1: Activity sequence for learning when to intervene human subject experiment

survey on their general puzzle solving habits. Figure E.1 illustrates the activity sequence of the experiment. 136 participants completed the study. The sample comprised of college undergraduate and graduate students in Computer Science, Psychology, Agriculture and Business majors. 117 of the 136 participants also completed the demographics survey.

E.3 Demography Survey Findings

Majority of the participants (39) were below the age of 20, while 38 subjects were between the age 20-25. Maximum age was 41 years. 70 of the 117 participants were male. When asked if they liked puzzle solving tasks, 78% of the participants either agreed or strongly agreed with the statement. Specifically to the Rush Hour game 79% of the participants liked or strongly liked Rush Hour. The most common reason as to why the participants liked puzzle solving tasks was that puzzle solving stimulates critical thinking skills. 30% of the participants usually did a puzzle solving task once a month, while 21% of the participants solved a puzzle once a week. When asked about strategies the participants used to solve difficult puzzles, 79% of the group said that they kept trying until the puzzle was eventually solved, while 12% of the participants said that they would ask for help.

Given a new puzzle that they have not seen before, if they get stuck while solving the puzzle, 26% said that they would not like any outside help. 47% of the participants said that they would like a suggestion/tip that would get them past the current situation. 15% said that they would like a warning, which indicated that their current approach would lead to a dead-end. 8% of the participants said that they would like a warning and an explanation to help them prevent getting stuck in the future. The most common medium for solving puzzles was using their mobile devices (42%). 31% of the participants used the personal computers/laptops to solve puzzles. 19% of the participants solved puzzles using physical means (e.g., puzzle books, newspapers and physical puzzles such as Rubik's cubes).

E.4 User Solutions Grouped by Length

For each puzzle, we sorted the solutions by the number of moves (in the complete solution) in the ascending order and split them into three groups (fast, medium, slow). We ensured that the three groups for each puzzle contained approximately equal number of users. Table E.1 summarizes the findings. There were 46 users in the fast group, 42 in the medium and 48 in the slow group. Mean refers to the mean number of moves in a solution produced by users who solved a specific puzzle. Forbidden moves refers to the number of times, the users who solved a specific puzzle moved the forbidden vehicle.

Table E.1: Plans produced by human users grouped by the mean number of moves and the number of forbidden moves

PID	Fast (46)		Medium (42)		Slow (48)	
	Mean	Forbidden Moves	Mean	Forbidden Moves	Mean	Forbidden Moves
P1	25.5	0	41.7	0	64.7	0
P2	74.7	4	137.7	16	277	28
P3	26.5	8	36.2	9	43.7	6
P4	25.2	1	32	4	76	28
P5	18.3	3	26	2	44.75	13
P6	22	0	24.5	0	39.6	0
P7	38.5	5	53.3	16	120	37
P8	9	0	9	0	10	0
P9	27.8	8	48.6	12	99	28
P10	50.3	11	66	14	127.3	46

Appendix F

Interactive Human-aware Intervention

We present the raw data for the usage of hints during the Interactive Human-aware Intervention human subject experiment and the evaluation metric values obtained for the control and the condition groups.

Table F.1: Users who saw the forbidden car alert message as the first hint. PID is the planning task identifier. percent complete $= \frac{\text{moves until the forbidden alert}}{\text{number of moves}}$. Threshold indicates the number of moves from the start during which the Human-aware Intervention agent did not function

User	PID	Total Number of Moves	Moves Until Forbidden Alert	Percent Complete	Threshold Number of Moves
1	P9	58	1	0.02	21
2	P3	106	16	0.15	25
3	P5	41	7	0.17	14
4	P2	81	22	0.27	30
5	P10	53	5	0.09	24
6	P13	55	1	0.02	21
7	P5	45	12	0.27	14
8	P3	47	16	0.34	25
9	P10	59	16	0.27	24
10	P6	62	19	0.31	22
11	P11	95	10	0.11	26
12	P3	93	14	0.15	25
13	P9	36	1	0.03	21
14	P5	34	8	0.24	14
15	P11	50	4	0.08	26
16	P5	40	6	0.15	14
17	P7	44	4	0.09	21
18	P3	121	19	0.16	25
19	P4	61	25	0.41	23
20	P4	60	23	0.38	23
21	P3	43	14	0.33	25
22	P5	40	6	0.15	14
23	P4	37	8	0.22	23
24	P2	111	24	0.22	30
25	P13	30	1	0.03	21
26	P4	78	10	0.13	23
27	P3	56	22	0.39	25
28	P10	69	18	0.26	24
29	P13	50	3	0.06	21
30	P2	189	20	0.11	30
31	P9	22	4	0.18	21
32	P2	172	42	0.24	30
33	P5	45	15	0.33	14
34	P5	38	6	0.16	14
35	P3	57	24	0.42	25
36	P3	71	20	0.28	25
37	P2	88	7	0.08	30
38	P4	37	9	0.24	23
39	P9	28	1	0.04	21
40	P3	64	15	0.23	25
41	P4	87	6	0.07	23
42	P10	60	23	0.38	24
43	P10	74	12	0.16	24
44	P4	49	7	0.14	23
45	P7	57	4	0.07	21
46	P10	46	13	0.28	24
47	P4	60	6	0.10	23
48	P4	27	5	0.19	23
49	P9	61	1	0.02	21
50	P9	71	2	0.03	21
51	P3	69	19	0.28	25
52	P10	59	5	0.08	24
53	P9	30	12	0.40	21
54	P7	195	22	0.11	21
55	P2	67	15	0.22	30
56	P13	35	1	0.03	21
57	P3	49	15	0.31	25

Table F.2: Mean number of remaining moves for the first ten hint requests for the Rush Hour type C planning tasks: P2, P4, P6 and P8

P2										
	Request 1	Request 2	Request 3	Request 4	Request 5	Request 6	Request 7	Request 8	Request 9	Request 10
Undo	96	-	90	54	0	379	375	374	366	320
Remaining	387	-	-	-	-	-	-	-	-	-
Next Best	99	145	90	54	59	44	43	42	41	40
Must Move	-	98	50	-	-	-	-	-	-	-
Restart	-	-	97	-	-	-	-	-	-	-
Ignore	-	96	95	172	383	-	-	-	-	-
P4										
Undo	44	70	-	-	-	-	-	-	-	-
Remaining	-	-	35	-	-	33	34	-	-	-
Next Best	31	34	30	41	23	22	21	25	27	24
Must Move	-	12	50	9	32	-	30	-	-	-
Restart	-	33	-	49	59	-	-	-	-	-
Ignore	-	12	11	42	22	7	-	-	16	22
P6										
Undo	43	-	-	-	-	-	-	-	-	-
Remaining	-	-	-	-	-	-	-	-	3	2
Next Best	-	24	9	8	-	-	5	-	-	-
Must Move	11	-	-	-	7	6	-	4	-	-
Restart	22	-	-	-	-	-	-	-	-	-
Ignore	-	-	37	-	-	-	-	-	-	-
P8										
Undo	-	-	-	-	-	-	-	-	-	-
Remaining	-	-	-	-	-	-	–	-	-	-
Next Best	4	3	-	-	-	-	-	-	-	-
Must Move	-	-	-	-	-	–	-	-	-	-
Restart	-	-	-	-	-	-	-	-	-	-
Ignore	-	-	2	1	-	-	-	-	-	-

Table F.3: Mean number of remaining moves for the first ten hint requests for the Rush Hour type E planning tasks: P1, P3, P5, P9 and P10

	Request 1	Request 2	Request 3	Request 4	Request 5	Request 6	Request 7	Request 8	Request 9	Request 10
P1										
Undo	-	-	-	-	-	-	-	-	-	-
Remaining	-	-	29	-	-	-	-	-	-	-
Next Best	32	30	33	34	7	6	5	-	-	-
Must Move	-	-	-	-	-	-	-	-	-	-
Restart	-	-	-	30	-	-	-	-	-	-
Ignore	40	41	-	-	-	-	-	4	3	0
P3										
Undo	53	59	-	26	-	-	-	-	-	-
Remaining	60	44	42	61	-	9	-	-	-	-
Next Best	25	31	33	26	24	30	26	23	24	21
Must Move	-	-	58	-	-	16	-	-	-	-
Restart	-	94	68	42	56	-	-	-	-	-
Ignore	-	42	36	31	24	18	23	24	13	-
P5										
Undo	32	31	-	-	34	-	-	-	-	-
Remaining	24	138	21	-	-	-	-	-	-	-
Next Best	20	22	20	81	21	20	19	18	11	16
Must Move	-	23	-	-	-	-	-	-	-	-
Restart	42	31	-	20	-	-	-	-	-	-
Ignore	74	20	47	19	23	26	-	-	22	-
P9										
Undo	44	63	-	-	-	-	-	-	-	-
Remaining	-	-	-	24	-	-	-	-	-	-
Next Best	-	38	36	35	34	33	32	31	30	29
Must Move	-	-	-	-	-	-	-	-	-	-
Restart	-	-	-	-	23	-	-	-	-	-
Ignore	-	-	25	-	-	-	-	-	-	-
P10										
Undo	47	27	-	-	-	-	-	-	-	-
Remaining	-	33	-	-	-	-	-	-	-	-
Next Best	-	27	28	24	23	19	28	27	26	25
Must Move	-	-	23	-	-	-	-	-	-	-
Restart	-	41	-	-	30	-	-	-	-	-
Ignore	22	41	22	19	20	19	-	-	-	-

Table F.4: Mean number of remaining moves for the first ten hint requests for the Rush Hour type M planning tasks: P7, P11, P12 and P13

	P7									
	Request 1	Request 2	Request 3	Request 4	Request 5	Request 6	Request 7	Request 8	Request 9	Request 10
Undo	89	-	27	-	-	-	-	-	107	-
Remaining	-	172	3	6	3	4	-	-	-	-
Next Best	7	20	66	95	14	93	16	-	14	13
Must Move	37	8	-	-	169	-	-	-	-	-
Restart	-	34	-	-	-	-	-	-	-	-
Ignore	-	-	35	34	-	9	59	61	-	-
	P11									
Undo	66	56	-	-	-	33	-	-	-	-
Remaining	-	-	26	-	-	-	-	-	-	-
Next Best	-	27	37	22	21	20	42	41	40	39
Must Move	32	-	-	-	-	-	-	-	-	-
Restart	-	-	-	-	-	-	-	-	-	-
Ignore	-	-	-	51	50	-	-	-	-	-
	P12									
Undo	-	-	-	-	-	-	-	-	-	-
Remaining	-	-	-	-	-	-	-	-	-	-
Next Best	-	9	8		6	-	-	-	-	-
Must Move	15	-	-	7	-	-	-	-	-	-
Restart	-	-	-	-	-	-	-	-	-	-
Ignore	-	-	-	-	-	5	4	-	-	-
	P13									
Undo	41	42	24	25	22	27	32	29	-	-
Remaining	12	-	3	-	14	-	-	-	-	-
Next Best	7	11	13	12	8	13	-	11	18	17
Must Move	-	-	-	-	-	-	-	-	-	-
Restart	-	-	-	-	24	-	-	-	-	-
Ignore	10	8	9	2	1	7	6	-	-	-

Table F.5: Latest time the landmarks are achieved in number of moves for the solutions produced by an automated planner and the human subjects who solved the planning task P1

	Landmark Id													
	L1	L2	L3	L4	L5	L6	L23	L22	L14	L13	L24	L16	L15	L17
Planner	3	4	5	6	7	7	23	22	14	13	24	16	15	17
0ED65A48A0	2	3	4	5	12	6	23	22	7	1	24	16	8	17
60FDED5C54	18	21	50	51	59	55	79	78	56	54	80	70	57	73
8A59F99069	31	32	33	34	36	38	55	54	39	35	56	48	40	49
88C193253A	6	32	33	34	1	38	50	49	39	37	51	42	41	43
2A7F3CC18D	1	3	4	5	13	6	37	36	22	21	38	26	25	27
9CA55504A0	30	33	34	40	14	41	51	50	42	38	52	44	43	45

Table F.6: The number of times each landmark is regained during the solution produced by the human subjects who solved the planning task P1

	Landmark Id													
User	L1	L2	L3	L4	L5	L6	L23	L22	L14	L13	L24	L16	L15	L17
0ED65A48A0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
60FDED5C54	2	2	3	3	2	2	2	2	2	2	1	2	1	3
8A59F99069	2	2	2	2	2	2	2	2	2	3	1	1	3	1
88C193253A	1	2	2	2	1	2	2	2	2	2	1	1	1	1
2A7F3CC18D	1	1	1	1	1	1	1	1	2	2	1	1	1	1
9CA55504A0	2	2	2	2	1	2	2	2	2	2	1	1	2	1

Table F.7: Latest time the landmarks are achieved in number of moves for the solutions produced by an automated planner and the human subjects who solved the planning task P2

	Landmark Id												
	L1	L30	L5	L6	L10	L32	L7	L31	L8	L25	L26	L29	L28
Planner	5	30	5	6	10	27	8	26	9	25	26	29	28
8A3F5735FF	166	190	143	167	176	63	171	153	175	182	188	186	189
8D843DCF85	129	174	130	148	163	152	132	146	161	164	165	160	166
6773F93121	81	112	90	91	98	107	92	106	96	104	110	108	111
9FF52D67E8	61	89	46	62	65	49	63	1	64	80	82	88	83
F776AEF9AB	47	68	6	48	54	63	49	62	53	60	66	64	67
1FDFF7AC44	404	426	394	406	412	421	407	420	411	415	424	422	425
1402D73F90	59	82	25	60	68	4	63	1	67	74	80	78	81
E1F19B9461	107	131	82	108	126	123	112	122	125	127	129	124	130

Table F.8: The number of times each landmark is regained during the solution produced by the human subjects who solved the planning task P2

	Landmark Id												
User	L1	L30	L5	L6	L10	L32	L7	L31	L8	L25	L26	L29	L28
8A3F5735FF	1	1	5	7	6	1	9	2	7	3	1	2	1
8D843DCF85	1	1	5	5	3	3	5	2	4	1	1	3	1
6773F93121	1	1	5	5	3	3	3	3	3	3	1	3	1
9FF52D67E8	1	1	3	4	2	2	2	1	2	2	1	5	1
F776AEF9AB	1	1	1	2	2	2	2	2	2	1	1	2	1
1FDFF7AC44	1	1	14	16	9	6	12	5	10	6	1	2	1
1402D73F90	1	1	2	3	2	1	2	1	2	2	1	2	1
E1F19B9461	2	1	6	7	5	3	6	3	6	2	1	3	1

Table F.9: Latest time the landmarks are achieved in number of moves for the solutions produced by an automated planner and the human subjects who solved the planning task P4

	Landmark Id				
	L2	L3	L21	L23	L22
Planner	3	4	21	23	22
9D60146003	1	2	23	25	24
81F7253198	41	44	77	79	78
EA49D74485	54	55	76	78	77
5E63E3F41A	21	22	59	61	60
D8E9B03AEA	29	36	59	61	60
5D801D87B4	1	2	60	62	61
B72E231048	31	32	87	89	88
C00FB34570	28	29	48	50	49
65DAA1BD3C	2	3	36	38	37
E53ED6E96E	1	7	26	28	27
A11273007C	21	22	36	38	37
802B116E16	2	8	32	34	33
1917B3DAA5	1	2	25	27	26
DF981E4FEB	1	14	27	29	28

Table F.10: The number of times each landmark is regained during the solution produced by the human subjects who solved the planning task P4

	Landmark Id				
User	L2	L3	L21	L23	L22
9D60146003	1	1	1	1	1
81F7253198	2	3	1	1	1
EA49D74485	3	3	1	1	1
5E63E3F41A	2	2	1	1	1
D8E9B03AEA	2	2	1	1	1
5D801D87B4	1	1	1	1	1
B72E231048	3	3	1	1	1
C00FB34570	2	2	1	1	1
65DAA1BD3C	1	1	1	1	1
E53ED6E96E	1	1	1	1	1
A11273007C	2	2	1	1	1
802B116E16	1	2	1	1	1
1917B3DAA5	1	1	1	1	1
DF981E4FEB	1	1	1	1	1

Table F.11: Latest time the landmarks are achieved in number of moves for the solutions produced by an automated planner and the human subjects who solved the planning task P6

	Landmark Id										
	L16	L15	L1	L2	L3	L4	L21	L20	L22	L14	L13
Planner	16	15	2	12	3	4	21	20	22	14	13
2CCC425959	54	53	22	6	43	44	62	61	63	52	45
8CABBDE04E	30	29	20	6	21	22	37	36	38	24	23
43F9EF2E21	17	16	3	1	4	5	23	22	24	7	6
752BC37F0E	17	16	2	4	3	6	24	23	25	15	7
36E7258DD8	18	17	3	1	4	5	27	26	28	16	6
658D25A080	38	37	25	24	26	27	44	43	45	36	28
6C5A7EE30F	15	14	1	3	2	4	21	20	22	13	5
49F9D4E78F	19	18	25	5	28	6	33	30	34	17	7
FA5C924928	27	26	12	6	14	15	34	33	35	17	16

Table F.12: The number of times each landmark is regained during the solution produced by the human subjects who solved the planning task P6

	Landmark Id										
User	L16	L15	L1	L2	L3	L4	L21	L20	L22	L14	L13
2CCC425959	1	1	1	1	1	1	3	3	1	1	1
8CABBDE04E	1	1	1	1	1	1	2	2	1	1	1
43F9EF2E21	1	1	1	1	1	1	1	1	1	1	1
752BC37F0E	1	1	1	1	1	1	1	1	1	1	1
36E7258DD8	1	1	1	1	1	1	1	1	1	1	1
658D25A080	2	2	2	2	3	3	1	1	1	2	3
6C5A7EE30F	1	1	1	1	1	1	1	1	1	1	1
49F9D4E78F	1	1	2	1	3	1	2	1	1	1	1
FA5C924928	1	1	1	1	1	1	2	2	1	1	1

Table F.13: Latest time the landmarks are achieved in number of moves for the solutions produced by an automated planner and the human subjects who solved the planning task P7

	Landmark Id													
	L1	L2	L3	L4	L5	L6	L10	L21	L7	L20	L9	L11	L18	L19
Planner	4	7	5	6	8	8	10	21	9	20	9	13	18	19
EE81806F5A	171	142	172	176	179	161	181	197	162	196	180	193	192	194
4833E5FF23	6	1	27	29	30	9	32	45	10	44	31	33	42	43
2769B127DC	16	23	17	45	46	24	48	59	33	58	47	49	56	57
563D8B218B	1	6	2	3	4	7	9	22	11	21	5	19	18	20
756F7DB232	9	1	10	13	14	2	16	31	3	30	15	17	28	29
BE69ABDC80	7	2	8	11	12	3	14	27	4	26	13	15	24	25
CA8A9046B0	25	27	34	35	36	42	38	59	43	58	37	39	56	57

Table F.14: The number of times each landmark is regained during the solution produced by the human subjects who solved the planning task P7

	Landmark Id													
User	L1	L2	L3	L4	L5	L6	L10	L21	L7	L20	L9	L11	L18	L19
EE81806F5A	8	3	7	8	8	4	3	1	6	1	8	4	6	1
4833E5FF23	1	1	2	2	2	1	1	1	1	1	2	1	3	1
2769B127DC	1	2	1	2	3	2	2	1	3	1	2	2	3	1
563D8B218B	1	1	1	1	1	1	1	1	1	1	1	1	1	1
756F7DB232	1	1	1	1	1	1	1	1	1	1	1	1	2	1
BE69ABDC80	1	1	1	1	1	1	1	1	1	1	1	1	2	1
CA8A9046B0	2	2	2	2	2	2	1	1	2	1	2	1	3	1

Table F.15: Latest time the landmarks are achieved in number of moves for the solutions produced by an automated planner and the human subjects who solved the planning task P8

	Landmark Id					
	L1	L5	L6	L7	L8	L9
Planner	1	5	6	7	8	9
B538F2D64C	5	3	6	7	8	9
7C7475AFF6	5	2	6	7	8	9
5B02E63E95	1	3	6	7	8	9
64C02661E5	7	10	11	12	13	14
B1640D5588	3	2	6	7	8	9
9FBF05CEC9	1	3	6	7	8	9
86CD8F3D28	5	2	6	7	8	9
CC65B278A9	5	2	6	7	8	9
6FBFA33753	3	2	6	7	8	9

Table F.16: The number of times each landmark is regained during the solution produced by the human subjects who solved the planning task P8

	Landmark Id					
User	L1	L5	L6	L7	L8	L9
B538F2D64C	1	1	1	1	1	1
7C7475AFF6	1	1	1	1	1	1
5B02E63E95	1	1	1	1	1	1
64C02661E5	1	1	1	1	1	1
B1640D5588	1	1	1	1	1	1
9FBF05CEC9	1	1	1	1	1	1
86CD8F3D28	1	1	1	1	1	1
CC65B278A9	1	1	1	1	1	1
6FBFA33753	1	1	1	1	1	1

Table F.17: Latest time the landmarks are achieved in number of moves for the solutions produced by an automated planner and the human subjects who solved the planning task P9

	Landmark Id																	
	L1	L2	L3	L4	L5	L6	L10	L21	L7	L20	L8	L9	L14	L16	L15	L18	L17	L19
Planner	2	3	4	5	6	7	10	21	8	20	9	14	14	18	17	19	18	19
E71D2E46E0	38	39	40	41	48	49	52	63	50	62	51	47	56	60	57	61	58	59
01C9FBDD1D	38	39	40	42	43	44	48	59	45	58	47	3	54	52	55	53	56	57
8BAF28A2FC	1	2	3	6	7	8	11	23	9	22	10	18	15	16	19	17	20	21
EE70D41DF8	6	7	8	9	15	16	19	31	17	30	18	26	25	23	27	24	28	29
51DF9C4A10	12	13	14	15	16	17	20	31	18	30	19	1	24	28	25	29	26	27
A3809304FB	8	9	10	11	12	14	17	29	15	28	16	3	22	26	23	27	24	25
E8BBAA27C4	52	53	54	55	56	58	61	73	59	72	60	51	65	66	69	67	70	71
305ED31CDD	28	29	30	31	32	34	37	49	35	48	36	1	41	42	45	43	46	47
38F4766DE8	27	9	10	11	12	14	17	37	15	36	16	3	30	34	31	35	32	33

Table F.18: The number of times each landmark is regained during the solution produced by the human subjects who solved the planning task P9

	Landmark Id																	
User	L1	L2	L3	L4	L5	L6	L10	L21	L7	L20	L8	L9	L14	L16	L15	L18	L17	L19
E71D2E46E0	1	1	1	2	2	1	1	1	1	2	1	3	3	2	2	2	2	2
01C9FBDD1D	1	1	1	2	2	1	1	1	1	2	1	1	2	2	2	2	2	2
8BAF28A2FC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
EE70D41DF8	1	1	1	1	1	1	1	1	1	1	1	2	2	1	2	1	1	1
51DF9C4A10	1	1	1	1	1	1	1	1	1	1	1	1	2	1	2	1	1	1
A3809304FB	1	1	1	1	1	1	1	1	1	1	1	1	2	1	2	1	1	1
E8BBAA27C4	2	1	1	3	3	1	1	1	1	2	1	2	2	2	2	2	2	2
305ED31CDD	1	1	1	1	1	1	1	1	1	2	1	1	2	2	2	2	2	2
38F4766DE8	2	1	1	1	1	1	1	1	1	1	1	1	3	2	2	2	1	1

Table F.19: Latest time the landmarks are achieved in number of moves for the solutions produced by an automated planner and the human subjects who solved the planning task P10

	Landmark Id								
	L15	L18	L17	L1	L3	L21	L23	L22	L14
Planner	15	18	17	21	15	22	24	23	14
8487A4F67A	63	66	65	67	34	68	70	69	60
B7624D7774	52	55	54	58	57	59	61	60	49
59E4C8FB8E	38	42	41	45	10	46	48	47	36
B76828B194	70	73	72	68	39	69	75	74	65
D50CD07A0B	43	45	44	42	1	46	48	47	39
155852E072	46	48	47	51	25	52	54	53	44
EE2F19282A	51	54	53	57	33	58	60	59	50
29AD5E25AE	53	56	55	57	39	58	60	59	49

Table F.20: The number of times each landmark is regained during the solution produced by the human subjects who solved the planning task P10

	Landmark Id								
User	L15	L18	L17	L1	L3	L21	L23	L22	L14
8487A4F67A	2	2	2	3	2	3	1	1	2
B7624D7774	2	3	3	4	2	3	1	1	2
59E4C8FB8E	2	2	2	2	1	2	1	1	2
B76828B194	3	2	2	3	2	3	1	1	3
D50CD07A0B	1	1	1	3	1	3	1	1	1
155852E072	2	2	2	2	1	2	1	1	2
EE2F19282A	2	2	2	2	2	1	1	1	3
29AD5E25AE	2	1	1	3	3	2	1	1	2