Supporting an Online Investigation of User Interaction with an XAIP Agent

Alan Lindsay,[1] **Bart Craenen,**[1] **Sara Dalzel-Job,**[2] **Robin L. Hill,**[2] **Ronald P. A. Petrick**[1]
[1] Department of Computer Science, Heriot-Watt University, Edinburgh, Scotland, UK
[2] School of Informatics, University of Edinburgh, Edinburgh, Scotland, UK
{alan.lindsay,b.craenen,r.petrick}@hw.ac.uk, {sdalzel,r.l.hill}@ed.ac.uk

Abstract

Human interaction relies on a wide range of signals, including non-verbal cues. In order to develop effective Explainable Planning (XAIP) agents it is important that we understand the range and utility of these communication channels. Our intention is to develop an interactive agent, whose behaviour is conditioned on the affective measures of the user (i.e., explicitly incorporating the user's affective state within the planning model). Accurate prediction of user affective state relies on real-time analysis of various predictors, which can require specialist equipment and calibration. However, the worldwide COVID-19 lockdown has meant that many intended lab-based experiments have now been moved online, making such real-time analysis impractical. As a result, we have developed a website to support a data gathering experiment, including a video stream (for facial expression analysis) with access to mouse positions and task performance, providing rich observations of the users as they interact with the agent and its plan. Underlying this system is the agent's behaviour strategy, which must be computed in advance and captured efficiently. This paper describes the built system and the challenges we faced getting it ready for deployment.

Introduction

A common approach to explanation generation in automated planning has been to represent the problem as one of model reconciliation (Chakraborti et al. 2017). In this way, a balance can be made between generating explanations that update the user's model of the environment and selecting explicable action sequences, which need no further explanation. However, at the heart of this approach is the requirement of an accurate user model, which is not practical in many applications. An alternative view is to instead see explanations within the wider context of human or agent interaction.

Humans adopt a wide range of communicative channels during interaction, which include both verbal and non-verbal modalities. Our intention is to examine human behaviour as they participate in a joint task with a computer agent. In a previous study (Petrick, Delzel-Job, and Hill 2019; Dalzel-Job, Hill, and Petrick 2020), we investigated measures of confidence in the context of an instruction-giving task. Biometric and behavioural measures (e.g., facial expressions and galvanic skin response) were analysed in conjunction with subjective, self-report measures of confidence combined with additional perceptions of a virtual human during the interactions. The relationships between variables could be utilised to predict users' confidence in, and levels of trust towards, a virtual human during an interaction and used to identify instances of confusion, suggesting an opportunity for providing extra information, such as explanations. Our intention was to build on this study to incorporate such real-time observations within the planning model, allowing the system to respond appropriately.

However, the worldwide COVID-19 lockdown has led to lab-based user studies being impossible. As such, we decided to adapt our approach and conduct a web-based user study, where the participants would be asked to interact with a system through a website. In order to support this approach, we have had to develop a website that is capable of generating the appropriate stimulus, as well as capturing the user's responses during the interaction.

In this paper, we present the developed system, which is currently being finalised in preparation of deployment. A key aspect of the work involves capturing the agent's behaviour: both its strategy for completing the task and its interaction actions (i.e., instructions and explanations). This has required balancing the competing requirements of task size (as a partially observable planning task), with user flexibility and limited size requirements (due to competition on bandwidth with other aspects of the system). We also conducted a feasibility study to compare alternative planning approaches for generating the agent's strategy. We present an empirical analysis and analyse the approaches, considering their appropriateness in the context of a real-time system.

We first present an overview of the intended user study and the system that will support it. We then present the agent and its underlying planning model and subsequently describe how its strategy is captured. We describe the representation of explanations and present some of the challenges of developing the system. We present our empirical evaluation, related work, and finally conclude.

The User Study

We are interested in investigating the range of human social signals, affective responses and behavioural patterns exhibited during co-operative joint action in a shared audio-visual environment. In this work, we consider an instruction giving and following scenario, adapted from the HCRC Map

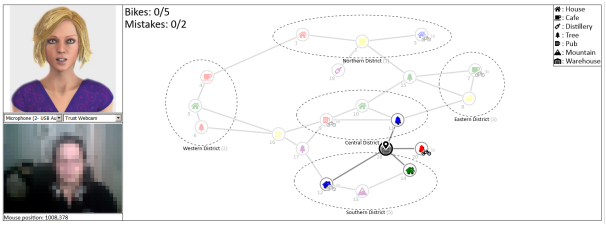

Figure 1: The user interface for the Bike Sharing Task. The user is able to see where the districts are positioned, as well as nearby landmarks. (Note: greyed out landmarks and roads would not be presented to a participant.)

Task (Anderson et al. 1991)[1] and the GIVE Challenge (Byron et al. 2009). In the Map Task, an instruction giver guides an instruction follower around a map using landmarks, while in the GIVE challenge the instructions are generated by a computer. The former task was designed to investigate human linguistic behaviour while the latter was proposed as a challenge for testing approaches to natural language generation. The importance of situating interactions within such tasks means that they are focused towards a common goal, and can inform on naturalistic behaviour in a task-based setting. As a result, our task has been designed to allow human-agent interactions to be observed in various scenarios typical of joint-action tasks, such as uncertainty as to an instruction's intent, and knowledge differences between the instruction giver and follower. These scenarios are not presented to the user in isolation, but as part of a complete system. We can therefore observe how a user responds to a particular strategy, and how their responses vary as a task progresses.

In this work, we underpin the agent's decisions and behaviour with a planning model that is used to generate the agent's strategy, as well as to construct situation-based instructions and explanations. Our intention is to investigate the relationships that exist between the information that the listener has been presented and their subsequent responses within specific situations. For example, how does the user respond to an ambiguous instruction and is their response different if they already know that the next target is, e.g., to the north? Understanding the human listener's needs and preferences during the interaction can inform our design of virtual agents that can meet and manage these requirements as they surface during the interaction.

The Bike Sharing Task

Our scenario involves a simple bike share company, which receives partial reports about the location of bikes. The virtual agent provides instructions to guide the human user around a map as they locate and collect the bikes. Figure 3 shows an example map, with the user at position X and the possible locations of a bike indicated with a dotted line.

[1] http://groups.inf.ed.ac.uk/maptask/

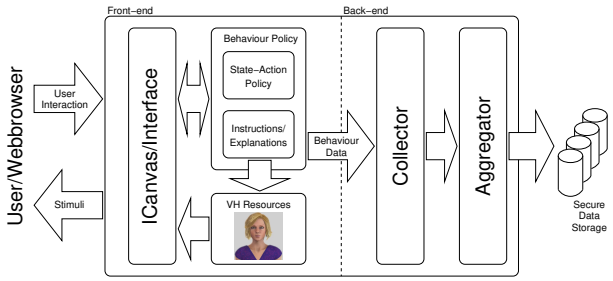

Figure 2: Overview of the system architecture.

System Overview

The system architecture, illustrated in Figure 2, consists of two main parts: a front-end website handling participant interactions; and a back-end server-based data collector and aggregator. The front-end of the website is developed using a combination of HTML5 and Javascript. The user interface (see Figure 1) includes an avatar for communicating with the participant, a frame indicating the data being collected, and an area depicting the interactive map.

The virtual instruction giver presents information to the human user using both verbal and non-verbal communication modalities. Its main function is to provide instructions and guidance on what the participant should do, e.g., move to a new landmark on the map, or pick up a bike. The virtual human avatar appears as in Figure 1, with behaviour prerecorded as a series, or combination of, videos that the system plays at predetermined times. Interaction with the user is defined by the avatar's behaviour policy, described below.

A portion of the screen (bottom left in Figure 1) is set aside to keep the participant informed on what data is being collected by the system. This frame displays the image stream, a representation of the sounds, and the mouse coordinates being recorded.

The main portion of the screen provides a visual representation of the interactive map on which the experiment plays out. The map presents a set of landmarks, with appropriate predefined icons and colours. Landmarks are connected to each other by roads, indicating which landmarks are currently available for the participant to move to. Only the current landmark, and its available neighbouring landmarks, are displayed, with the available landmarks updated after a participant's move. The bikes that are visible to the participant, either at the current landmark, or, in some cases, at the neighbouring landmarks, are depicted in the lower right-hand corner of the landmark. Movement across the map is indicated by clicking on a neighbouring landmark, picking up a bike by clicking on the current landmark when a bike is present there. In addition to landmarks, bikes, and roads, the map also depicts various districts (e.g., the Central or Northern District). Districts are included to give the avatar a reference for its guidance, so that it can provide instructions of the kind: 'We can now go looking for bikes in the Northern District'. Finally, the map includes a legend for explaining the meaning of the landmark icons, and two counters: one indicating the number of collected bikes and the other indi-

cating the number of mistakes made. The map layout and its constituent parts is determined by a map definition file, which is also used to generate the agent's planning model.

The back-end server-based part of the system provides a means for the system to collect and aggregate behavioural and interactive data from the participant for later offline study and analysis. Its primary functionality is to collect the video, audio, and mouse movement data, which are transmitted to a server-side collector, where they will be aggregated for each individual participant. In addition to these three data modalities, data about which and when landmarks or bikes are clicked, when and where bikes are picked up, and mistakes made are also collected. All data collected is time-stamped and aggregated for an individual user through the use of an anonymous hash or ID. Combined with the map and policy definition, the collected data will allow for a full analysis of how the system interacted with participants, and how individual participants interacted with the system.

The Agent's Planning Model

In this section, we first present the planning background followed by the agent's planning model which underpins the agent's behaviour. A partially observable planning problem, e.g., (Bonet and Geffner 2011), can be defined by a tuple, $P = \langle F, A, M, I, I^P, G \rangle$, with fluents F, actions A, sensor model M, the actual initial state I, the positive and negative literals of the state known by the agent $I^P \subseteq I$, and goal G. An action is defined by its preconditions and effects. An action is applicable if its preconditions are satisfied in the agent's partial state and the application of an action causes its effects to be applied to the agent's current state. Sensing actions are triggered whenever they become applicable and their observations update the agent's state. The set of potential agent (partial) states is represented by S^P and can be enumerated by expansion from the agent's initial state. A solution to the problem is a state-action policy, $\pi : S^P \mapsto A$, such that a simple executive can use the policy to iteratively step from the initial state to a state that satisfies the goal, by looking up the current state in the policy mapping and applying the selected action. We note that as our action models are deterministic we only require a partial mapping.

The agent's planning model captures a basic transportation style domain, including 'move' and 'pickup-up' actions and 'sense-bike' sensing actions. The bike locations are initially unknown. The partial reports (such as $bike_1$ is in the Western District) are used to partially constrain the possible initial states. The 'sense-bike' sensing actions are defined for particular bike and location pairs and determine whether the bike is at the location. The sensors are activated when the agent is at the location.

For the purposes of this study, the agent's model is built on the assumption that the agent and user have the same knowledge. As such, this model is used to generate the agent's strategy, which generates the next instruction at each step. For example, in the scenario illustrated in Figure 3, the next plan action might be (`move green_house`). This action is linked to a specific speech utterance, such as 'Go to the green house.' The model is also used to generate explanations, such as indicating the agent's intention (e.g., identifying which bikes will be looked for next).

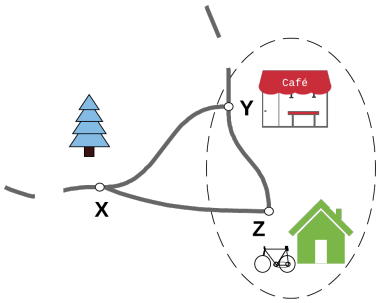

Figure 3: An example scenario starting at the tree (X), with 2 possible bike locations (Y or Z) indicated by a dotted line.

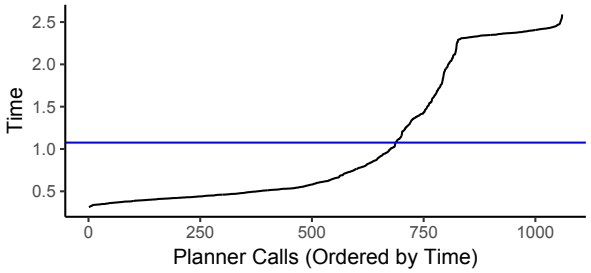

Figure 4: A graph indicating the length of time (in seconds) to run an external planner and parse the resulting plan. Planner calls are sorted by their time (to improve readability) with the blue line indicating the mean.

Offline Online Planning

Our intent is to use our findings in this study to ground our future work in XAIP. It is therefore essential that we use a planner to underpin the agent's interactions. One exciting option was to use the 'planning.domains' website (Muise 2016), which provides access to a planner in the cloud. Although it does not natively support partial observability, it does provide a classical planner that would be sufficient for a compilation-based approach, such as (Bonet and Geffner 2011). However, it is important that we retain some basic guarantees over planning performance. In particular, unpredictible behaviour (e.g., time outs or a busy server) and its consequences could distract from the key aspects of our current study. We therefore construct a policy in advance from our planning model, which is used to control the interaction with the user as they complete the task on the website.

Bringing Execution Forward

Partially-observable state spaces can be very large. As the network is navigated the state reflects the updated beliefs about the unknown part of the state. For example, if the user is currently at X in Figure 3 and moves to Y, then the state will be updated with the knowledge that the bike is not at Y. If, as in this example, Y and Z were the only options for the

Algorithm 1 BOUNDEDEXECUTION(BPE): Expanding a bounded exploration.

```
1:  function BPE(n_err, π^<, s)
2:      if s ⊨ g then
3:          Π ← Π + π^<
4:      end if
5:      π^> ← PLAN(s)
6:      for all a ∈ A do
7:          cost ← COST(a, π^>)
8:          if s ⊨ a ∧ n_err − cost > 0 then
9:              s' ← APPLY(s, a)
10:             BPE(n_err − cost, π^< + a, s')
11:         end if
12:     end for
13: end function
```

bike then it may also infer that the bike is at Z. Although it is important that we plan using a partially-observable model, it is clear it is not necessary to generate the complete contingency tree. We can instead exploit the fact that we know the true state in advance and can simulate the sensing actions in advance, effectively bringing the execution forward. This allows us to use an online partially-observable planning approach to compute the plan in advance and greatly reduce the number of states that we have to consider.

Of course, decoupling plan generation from the real-time constraint raises the possibility that the plans might not be generated within a realistic real-time replanning window. For our standard approach, a 3 seconds time-out was used on the underlying planner. This is on the upper end of what is reasonable for real-time performance, but it allowed us to be confident that the planner would return a plan. Figure 4 presents a graph plotting the time to run and process the plan (i.e., clock time in the framework rather than just planning time) during the generation of a policy. The average is a little over 1 second. The higher times typically coincide with changes in belief and deviations from the path. In each of these cases we might expect that any delay in calculating the next strategy would be mitigated by the agent acknowledging the 'unexpected' event. We therefore believe that such plans might reasonably be used for 'real-time' interactions.

User Exploration and Mistakes

The aim of the user study is to observe how human users respond while interacting with a computer controlled agent. An important aspect of this work is observing how users respond when the intentions of the agent are not clear or when they do something wrong. In each case, we would like users to be allowed to deviate from the agent's intended path. It was also our desire that the agent's strategy was guided by plans and that we did not resort to a repair strategy, which might have inadvertently impacted upon the user's experience. However, as we have observed above, the state space is large and even allowing a single error greatly increases the number of possible paths through the space. In order to address this concern we considered a limited and targeted amount of deviation from the intended path.

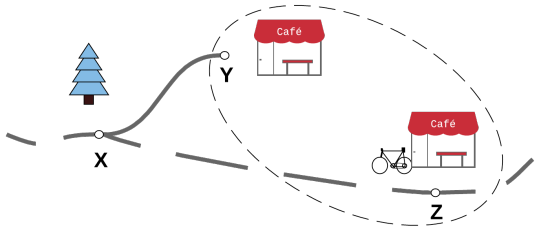

Figure 5: An example scenario starting at the tree (X), with 2 possible bike locations (Y or Z) indicated by a dotted line.

Errors	0	1	2	3
gt	31	142	550	1528
eq	31	172	3127	85566

Table 1: Policies were generated using Algorithm 1 extended with a store of visited states. States were matched when the stored state had equal (eq) and greater than (gt) number of errors remaining. The table presents the number of states in the resulting Bike Share policies with $0-3$ errors.

Allowing Errors We extended the online planner to allow a bounded exploration. The pseudocode is shown in Algorithm 1, simplified for the classical planning approach. The recursive algorithm generates a collection of plans (line 2). A plan is used to determine the agent's intended route (line 5). Each of the applicable actions is considered and if there are sufficient errors remaining (e.g., $n\_err > 0$) for that particular execution path then the action is followed and the number of errors is updated (line 10). At this stage, we assume that the function $\texttt{COST}(a, \pi)$ returns 0 if a coincides with the first plan action, and 1 otherwise.

The algorithm presented in Algorithm 1 is inefficient: we often will not need to generate a plan at every step and we will also potentially repeat states. In particular, as we explore the possible bounded executions we would like to exploit the plans that we have already generated when we rediscover a particular state. This can greatly reduce the size of the policy. However, it is important to record the number of errors that were remaining with each state that is stored. If a state was expanded with fewer errors then the set of execution paths that will have been explored from that state represents only a subset of those needed for the current execution node. If the state was expanded with more errors then it will have more execution paths and therefore could allow more errors. However, this can easily be addressed during execution (on the website) by counting the number of errors. Moreover, it can greatly compress the size of the policy (see Table 1).

Targeted Path Deviation Our investigation will attempt to examine human user response in situations of uncertainty, such as when users are given an ambiguous instruction or have a different knowledge level. As a result, there will be situations where the user might be unsure of the intended path and might be more likely to make a mistake. For example, in the example in Figure 5 starting from X the agent might produce the instruction 'Go to the cafe', which is clearly ambiguous. Conversely there will be situ-

```
<state>
    <id>32</id>
    <plan_action_id>15</plan_action_id>
    <allowed_transition>
        <action_id>15</action_id>
        <next_state_id>31</next_state_id>
        <is_mistake>0</is_mistake>
    </allowed_transition>
     ...
</state>
```

Figure 6: An example policy state indicating the state id, the possible actions and the resulting policy state.

```
<behaviour>
    <state_id>32</state_id>
    <basic_guidance>move_cafe.mp4</basic_guidance>
    <revisit_guidance>move_again_cafe.mp4</revisit_guidance>
    <further_guidance>indicate_red.mp4</further_guidance>
    <confused>target-k-east-bike.mp4</confused>
    <transition_behaviour>
        <action_id>15</action_id>
         <commentary>NONE</commentary>
    </transition_behaviour>
     ...
</behaviour>
```

Figure 7: A behavioural template defined for a policy state, indicating the agent's guidance and explanation.

ations where users know more than the agent and might be encouraged by the agent to use that knowledge to make a better action choice. For example, if the user can see the bike at Z then they might take the initiative to move there. It is therefore necessary to extend the explored execution paths so that these transitions are not penalised.

To achieve this, we have extended our system to accept a collection of permissibility rules of the form $a \mapsto b$. Each rule defines an ordered relationship between the two actions, a and b, which is interpreted as: if a is the intended action then b is also permissible. For example, in the case of the ambiguous instruction in Figure 5, and assuming that the agent was intending to move to Y, then we could include a rule indicating that it would be permissible for the user to move to Z instead. These rules are used during the calculation of the action cost in Algorithm 1 and extend the collection of actions that receive zero cost.

The State-Action Policy

The resulting state-action policy defines an action for each of the distinct states. An example state is presented in Figure 6. For our experiment we computed policies for 4 problems, each allowing 2 errors and > 20 permissibility rules. The policies had around 2300 states on average and one of the generated policies (with around 1700 states) captured the behaviour for over 9.0×10^9 possible execution traces.

Policy-Based Explanations

In this section we describe the instruction and explanation generation process. The selection of instructions and explanations that determine the behaviour of the virtual human are tied to the state-action policy. Each policy state is associated with a behaviour template. Each template determines

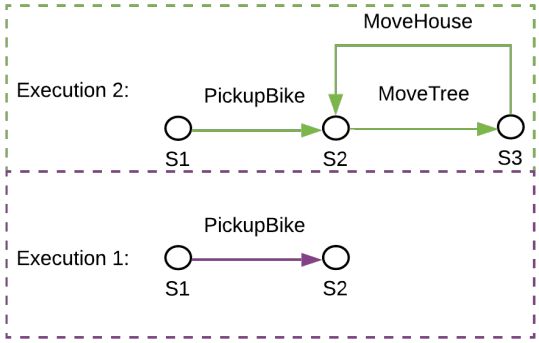

Figure 8: Two execution traces that result in the same policy state. The policy state S2 is visited twice in Execution 2.

the agent's behaviour while the execution is in the associated state. An example is presented in Figure 7. This template is defined for the example presented in Figure 3 and indicates that the agent should instruct the user to move to Y.

The template defines 3 main state-based dialogue actions: an initial dialogue action (e.g., for an instruction), a second dialogue action, which occurs after user hesitation (e.g., for elaboration or explanation) and a final dialogue action, if the user indicates that they are confused. In this way, we can capture the agent's instructions along with pre- and post-explanations based on the policy state. This context is sufficient for describing the current scenario and intent-based explanations. For example, the agent can use its plan to indicate the next subgoal it intends the user to achieve.

Beyond State-Based Dialogue Actions

As each state in the policy represents many possible execution states, state-based explanations are less effective for generating explanations of previous events. A simple example is presented in Figure 8, where two execution plots from position Z in Figure 3 are represented by the same policy state. In the first execution (indicated in purple) the user has arrived at the house and collects the bike. It would be expected that this progress towards achieving the goal is acknowledged by the agent. However, in the second execution, the user has picked up the bike, but then moved to the tree, before returning to the house. The execution returns to the policy state S2, but it would not be appropriate for the agent to comment again on the bike being picked up. Notice, this form of loop is allowed because the user can make errors (loops in the agent's strategy are obviously not allowed).

We can mitigate part of this issue by extending our template to allow dialogue actions to be associated with the selected state-action pairs. This is supported by our behaviour templates (Figure 7) through transition behaviours. E.g., we link the `pickup-bike` action with its acknowledgement.

Further extensions can only be approximated using this policy approach. E.g., only allowing the agent to comment on some feature when it is guaranteed in every execution trace. Otherwise, we must maintain supporting information during execution. As an example, consider a situation where the agent is directing the user to return to a previously visited

location. In this case, the agent might indicate that the location has already been visited, e.g., 'Go back to the house'. This might help to disambiguate instructions, or simply acknowledge that the agent knows that they have been there already. We have included an alternative dialogue action in the template for situations where the agent indicates a location that has already been visited, i.e., 'revisit_guidance'. This is supported by retaining a list of the visited landmarks.

Precompiling the Agent's Knowledge

Generating the behaviour policy for the agent in advance comes with certain guarantees that are difficult to replicate in a real-time setting, but it also requires considerable computational resources. The precompilation approach allows us to examine the policy in advance in order to be certain that expected properties hold, e.g., the agent's instructions do not contain loops and always lead to the goal. As the analysis is performed in advance we do not have to contend with processes not returning or any unexpected results during the execution. These are beneficial for protecting our experiment from distracting influences that are not part of our current study. The precompilation also allows us to consider approaches and processes that are currently time prohibitive for a real-time setting. This allows us the opportunity to examine the benefit of particular approaches and observe in advance whether it is worth investing effort to make them more efficient, or to investigate approximation methods.

Whereas these factors have given us a desired freedom and security, the precompilation approach also has its drawbacks. Defining the agent's behaviour at each of the policy states requires a considerable amount of precomputation. Depending on the intended deployment, it is likely that many of the states and transitions described in the policy will never be visited in any execution. Nonetheless, the policy must still be analysed for each type of explanation, just in case.

System Challenges

Because the system lacks an online behavioural analysis component, assumptions have had to be made on the affective state of the participant while it is interacting with the system. One challenge with this is how and when to assume that a participant is confused by the guidance given by the avatar. In the system, we assume that a participant is confused simply because the participant has not interacted with the system for a set, definable, period of time.

Another challenge, shared among online systems, is how to deal with the possibly concurrent nature of the participants responses. At any one time, more than one participant could be interacting with their individual instance of the system, which requires specific multi-user identifiable data to be collected and aggregation at the back-end server-side of the system. Given that the primary data modalities all require timely and (fairly) high-rate data collection, special care has to be taken to account for different throughput rates of the data, especially since no guarantees can be given as to the bandwidth availability of each user. Moreover, this works in both ways, in that individual participants also require access to a (possibly large) variety of guidance videos. Should the data transmission from the server to a participant's platform be obstructed somehow, even for a short period of time, the resulting interruption in the system's interaction with the participant may influence participant behaviour and/or evaluation of the system. To address this challenge, the system tries to pre-load all interaction components and maintain a cache, for example, of all interaction videos.

Problem\Planner	FF	Cloud	Lama
Map 1 ($\gamma = 1.0$)	34.00	27.00	27.00
($\gamma = .95$)	35.45 (2.09)	30.64 (11.10)	28.35 (1.84)
($\gamma = .90$)	36.87 (2.68)	32.62 (8.71)	29.27 (2.21)
Map 2 ($\gamma = 1.0$)	30.00	27.00	24.00
($\gamma = .95$)	31.78 (2.31)	∞	25.60 (1.80)
($\gamma = .90$)	33.02 (2.83)	∞	26.75 (2.24)
Map 3 ($\gamma = 1.0$)	32.00	31.00	23.00
($\gamma = .95$)	34.38 (3.47)	∞	24.61 (1.80)
($\gamma = .90$)	35.84 (4.02)	∞	25.89 (2.33)
Map 4 ($\gamma = 1.0$)	33.00	∞	31.00
($\gamma = .95$)	35.09 (2.04)	∞	32.76 (1.67)
($\gamma = .90$)	36.47 (2.40)	∞	33.81 (1.90)

Table 2: Sampled means (and standard deviations) of the execution length for 1000 policy samples and 4 maps, with γ controlling the probability of the user following the intended plan step.

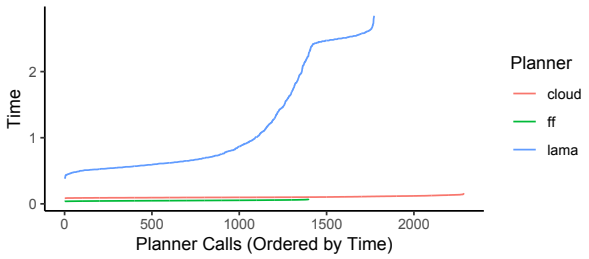

Figure 9: The call times to external planners (ordered by time) while generating the policies for Map 2.

Empirical Evaluation

As part of our preparations for conducting the online user study we have performed a feasibility study to examine possible alternative planning configurations. In this section, we present empirical results generated as part of this study. Our approach to partially-observable planning relies on a compilation to classical planning (Bonet and Geffner 2011), which

# goals	#States	Lama $\gamma = 1.0$	Lama $\gamma = .95$	Lama $\gamma = .90$
1 bike	59	8.0	8.60 (1.06)	9.33 (1.50)
2 bike	146	13.00	14.04 (1.31)	14.84 (1.62)
3 bike	485	21.0	22.48 (1.53)	23.71 (1.81)
4 bike	1159	22.0	23.73 (1.73)	25.09 (2.06)
5 bike	1762	27.0	28.35 (1.84)	29.27 (2.21)

Table 3: Policy state count and sampled means (and standard deviations) of the execution length for 1000 policy samples, for the goal sets ($1 - 5$ bikes) of Map 1, and probability of the user following the plan controlled with parameter γ.

supports efficient plan generation for partially-observable problems via replanning. Underpinning this approach is a classical planner, which we analyse using alternative planners to understand the trade-offs between time, accuracy and the required computing resources in the resulting policies.

In our user study, each participant will be presented with 4 conditions. As such, we constructed 4 maps each with around 20 landmarks and 5 bikes (similar to the graph in Figure 1). The goal of each problem was to find and collect the bikes and return to the base. We generated the policies to allow 2 execution errors as well as the permissibility rules described earlier, with each map describing over 20 permissibility rules. The planners we used in this study were:

FF The FF planning system (Hoffmann and Nebel 2001),

Cloud A server-based clone of the 'planning.domains' solver (Muise 2016),

Lama The LAMA-11 configuration of Fast Downwards (Richter and Westphal 2010) with a 3 second time-out.

It should be noted that the comparison of quality and time between these planners is not intended to be *fair* in a traditional sense, e.g., FF and Cloud both return their first plan, whereas Lama is being used as an anytime planner with a 3 second time-out. However, the intention is to gain an understanding of how these planner configurations compare in the context of a real-time system.

Policy Quality

In order to analyse the quality of the guidance captured in the policy we used each policy to generate execution samples. A parameter, γ, was used to control the probability that the simulated user would follow each instruction. Table 2 presents the mean plan lengths for the different planner configurations for $\gamma = \{1.0, 0.95, 0.90\}$. The results show that using Lama as the underlying classical planner resulted in policies with better quality. The quality of the FF policies were worse and resulted in often substantially longer executions. It is interesting to note that these plans often featured inefficiency at a local level (e.g., using 2 moves where 1 was possible) and a global strategy level (e.g., poor subgoal ordering). However, in both Lama and FF the plan lengths are relatively consistent between the samples. When using the policies generated by the Cloud approach the execution sampler frequently did not return. This was caused by redundant loops in the plans generated by the Cloud approach, resulting in execution loops in the captured policy. The Cloud approach generates a reasonable plan for the agent ($\gamma = 1.0$) for Maps 1, 2 and 3, but for Map 4 the agent's intended plan includes a loop. The sampled plan lengths (for $\gamma < 1.0$ in Map 1) have high variability. Although loops in the policy can be detected and removed during construction, we believe it is interesting to observe this issue as it also has implications for using this approach for real-time replanning.

To provide an indication of how the policies grow with the complexity of the problem we have generated policies for increasing subsets of goals for Map 1, from 1 to 5 bikes. Table 3 presents the number of states in each policy and the average plan lengths for different γ values. The results show that, unsurprisingly, as the problem size grows the policy also grows. It is interesting to note that plan length is not the only factor dictating policy size, e.g., there is a disproportionate increase in state counts between the policies for 3 and 4 bikes. Policy size is sensitive to the number of state clashes that occur when it is generated. This is dictated by both the consistency of the planners, as well as the number of alternative pathways between points in the search space. As more goals are added it becomes more likely that 'errors' will lead to different goal orderings and different plans.

Planner Reaction Time

In order to understand how the generated plans might be used within a real-time system, we have captured the call time to the classical planners from within the K-Replanner system (Bonet and Geffner 2011). These times therefore combine various processes, including parsing the resulting plan, and provide a better indication of the planner being used as a delegate within an actual real-time system. The call times captured during the generation of Map 2 are presented in Figure 9. The FF and Cloud configurations (the planners that return their first plans) generate plans far quicker than Lama. It is clear that the application will determine the appropriate configuration given the trade-off between time and quality. We noted earlier that Lama's times might be considered reasonable in certain contexts, e.g., interaction planning, where dialogue actions might take several seconds and provide time for planning in the background. One important additional consideration is that the Cloud approach could be used in a situation with limited computational resources.

Related Work

The increasing adoption of AI Planning in real world applications has led to a growing focus around Explainable Planning (XAIP) (Fox, Long, and Magazzeni 2017) and its key issues: co-operation, transparency, and trust. The representational requirements of explanations have previously been investigated in (Sohrabi, Baier, and McIlraith 2011; Vallati, McCluskey, and Chrpa 2018; Lindsay 2019). In (Sohrabi, Baier, and McIlraith 2011), they observe that selecting preferred explanations will often rely on the past. As we have seen, this is a key limitation of connecting explanations to policy states. However, their compilation approach would lead to the salient features of the past interaction being recorded within the policy states, making our approaches compatible. In (Miller 2019), it is argued that explainable AI (XAI, of which XAIP forms a part) should be based on the findings of previous work in the social sciences, such as cognitive science. Our aim is to bring together experimental research in cognitive science, involving cooperative joint action, with the practical construction of automated planning tools to apply to the task of explanation generation. Previous results already provide insight into preferred styles of explanation (Foster et al. 2009; Carletta et al. 2010; Henderson, Matheson, and Oberlander 2012). Our intention is to clarify and extend these results in the context of plan-based agent interaction.

Exploiting planners in order to manage user interactions is not new. For instance, (Petrick and Foster 2013) presented a robot bartender that could successfully balance multiple simultaneous customers in a task-based setting, using the knowledge-level PKS planner (Petrick and Bacchus 2002) to construct contingent plans conditioned on the customer's social states. Our current work intends to inform future work in these sorts of systems by improving our understanding of a broader range of human communication signals, and how they can be utilised to inform action selection.

Conclusion and Future Work

We intend to conduct a web-based user study to investigate human responses during interaction with a plan-based agent. We have developed a website, where the user performs a joint task with a computer agent based on a simple bike sharing scenario. The website also captures the user's behaviours and responses, including mouse position and a video stream for post-analysis. The agent's behaviour must be precomputed and represented, which is made more challenging as the user's responses and choices at each step are unknown in advance. We have described our approach, which links the agent's instructions and explanations with the states of their strategy. We presented an empirical analysis comparing 3 planning approaches, including a cloud-based planning approach, in terms of solution quality and reaction time, which we have used to select the most appropriate strategy for our study. After this study is complete and the data analysed, we hope to conduct a lab-based study. Our aim is to enhance our agent's world model using our improved understanding of human behaviour, in order to enable the agent to respond reactively to user signals, and harness the representational benefits of approaches like epistemic planning (Bolander 2017; Petrick and Bacchus 2002) in partially-observable domains.

Acknowledgements

This work is funded by the UK's EPSRC Human-Like Computing programme under grant number EP/R031045/1.

References

Anderson, A. H.; Bader, M.; Bard, E. G.; Boyle, E.; Doherty, G.; Garrod, S.; Isard, S.; Kowtko, J.; McAllister, J.; Miller, J.; et al. 1991. The HCRC map task corpus. *Language and speech* 34(4):351–366.

Bolander, T. 2017. A Gentle Introduction to Epistemic Planning: The DEL Approach. In *9th Workshop on Methods for Modalities*, 1–22.

Bonet, B., and Geffner, H. 2011. Planning under partial observability by classical replanning: Theory and experiments. In *International Joint Conference on Artificial Intelligence*.

Byron, D.; Koller, A.; Striegnitz, K.; Cassell, J.; Dale, R.; Moore, J. D.; and Oberlander, J. 2009. Report on the first NLG challenge on generating instructions in virtual environments (GIVE). In *12th European Workshop on Natural Language Generation*.

Carletta, J.; Hill, R. L.; Nicol, C.; Taylor, T.; De Ruiter, J. P.; and Bard, E. G. 2010. Eyetracking for two-person tasks with manipulation of a virtual world. *Behavior Research Methods* 42(1):254–265.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan explanations as model reconciliation: moving beyond explanation as soliloquy. In *26th International Joint Conference on Artificial Intelligence*, 156–163.

Dalzel-Job, S.; Hill, R. L.; and Petrick, R. P. A. 2020. Start making sense: Predicting confidence in virtual human interactions using biometric signals. In *Measuring Behavior*.

Foster, M. E.; Giuliani, M.; Isard, A.; Matheson, C.; Oberlander, J.; and Knoll, A. 2009. Evaluating description and reference strategies in a cooperative human-robot dialogue system. In *IJCAI*, 1818–1823.

Fox, M.; Long, D.; and Magazzeni, D. 2017. Explainable planning. *arXiv preprint arXiv:1709.10256*.

Henderson, M.; Matheson, C.; and Oberlander, J. 2012. Recovering from non-understanding errors in a conversational dialogue system. In *SeineDial: The 16th SemDial Workshop on the Semantics and Pragmatics of Dialogue*, 128.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Lindsay, A. 2019. Towards exploiting generic problem structures in explanations for automated planning. In *10th International Conference on Knowledge Capture*, 235–238.

Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* 267:1–38.

Muise, C. 2016. Planning.Domains. In *26th International Conference on Automated Planning and Scheduling, System Demonstrations*.

Petrick, R. P. A., and Bacchus, F. 2002. A knowledge-based approach to planning with incomplete information and sensing. In *AIPS*, 212–222.

Petrick, R. P. A., and Foster, M. E. 2013. Planning for social interaction in a robot bartender domain. In *ICAPS*, 389–397.

Petrick, R. P. A.; Delzel-Job, S.; and Hill, R. L. 2019. Combining cognitive and affective measures with epistemic planning for explanation generation. In *ICAPS 2019 Workshop on Explainable Planning (XAIP)*, 141–145.

Richter, S., and Westphal, M. 2010. The lama planner: Guiding cost-based anytime planning with landmarks. *Journal of Artificial Intelligence Research* 39:127–177.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. A. 2011. Preferred explanations: Theory and generation via planning. In *25th AAAI Conference on Artificial Intelligence*.

Vallati, M.; McCluskey, T. L.; and Chrpa, L. 2018. Towards explanation-supportive knowledge engineering for planning. In *The EXplainable AI Planning Workshop (XAIP-18)*.