Order Matters: Generating Progressive Explanations for Planning Tasks in Human-Robot Teaming

Mehrdad Zakershahrak, Shashank Rao Marpally, Akshay Sharma, Ze Gong, and Yu Zhang

Abstract— **Prior work on generating explanations in a planning context has focused on providing the rationale behind an AI agent's decision-making. While these methods offer the right explanations, they fail to heed the cognitive requirement of understanding an explanation from the explainee or human's perspective. In this work, we set out to address this issue by considering the order for communicating information in an explanation, or the *progressiveness of making explanations*. Progression is the notion of building complex concepts on simpler ones, which is known to benefit learning. In this work, we investigate a similar effect when an explanation is composed of multiple parts that are communicated sequentially. The challenge here lies in determining the order for receiving different parts of an explanation that would assist in understanding. Given the sequential nature, a formulation based on goal-based MDP is presented. The reward function of this MDP is learned via inverse reinforcement learning based on training data. We evaluated our approach in an escape-room domain to demonstrate its effectiveness. Upon analyzing the results, it revealed that the desired order arises strongly from both domain-dependent and independence features. This result confirmed our expectation that the process of understanding an explanation for planning tasks was progressive and context dependent. We also showed that the explanations generated using the learned rewards achieved better task performance and simultaneously reduced cognitive load. These results shed light on designing explainable robots across various domains.**

I. INTRODUCTION

As robots benefit a diverse set of domains, human-robot interaction has evolved to be an increasingly important subject. In human-robot teaming, it is desired that the interaction occurs coherently, as is observed in human-human teaming [1], [2]. Like with a human teammate, a robotic teammate is expected to understand its human partners and explain its decisions or behaviors when necessary. Explanations in a teaming context support the rationale behind a teammate's decision-making [3] and help maintain shared situation awareness and trust [4], [2]. Although there is much prior work on generating explanations, the focus has been on developing the right explanations from the explainer's perspective rather than good explanations for the explainee [5], [6], [7].

Unsurprisingly, the right explanation may not necessarily be a good explanation–anyone with teaching or mentoring experience would share such a sympathy. The dissonance between the explainer and explainee may result from various inconsistencies, such as information asymmetry or different cognitive capabilities, to name a few. These inconsistencies may be summarized as *model differences*–the differences between the cognitive models that govern the generation and interpretation of an explanation, respectively, for the explainer and explainee [8]. When these two models are the same, as is assumed in most prior work, an explanation from the explainer's perspective would be perfectly right and understandable to the explainee, as if the explanation were made for the explainer itself. The more general case where the models differ has also been investigated [9], [10] under the model reconciliation setting. The focus there is on explaining domain model differences such that the two models become more compatible for a given plan. However, one remaining challenge in explanation generation is to consider the differences in the cognitive capabilities for understanding an explanation.

In this work, we take a step further by generating explanations in a way that assists their understanding for the explainee. This is especially relevant in human-robot teaming since robots are frequently deployed to situations that require a high cognitive ability that humans often do not possess. In such cases, generating explanations that are easily understandable is equivalent to reducing the cognitive effort required for interpreting them. In this work, we study the order for communicating information in an explanation. Except for very simple domains, making an explanation is not an instantaneous effort; instead, information in an explanation must be conveyed in small parts sequentially. Explanation generation thus relates to learning progression in the psychology and education literature in observance of humans' cognitive limitations [11], [12], [13]. It is based on the intuition that complex concepts should be built on the understanding of simpler ones. The idea of *progressive explanation generation* draws inspiration from a similar problem characterization when explaining a plan: the interpretation of new information depends on the current context established as a result of the information received earlier. Consider the following example of a conversation between two friends, which illustrates the importance of providing information in a proper order when making an explanation:

```
Amy: Let's go to the outlet today.
Monica: My car is ready.
Monica: The rain will stop soon.
Amy: Wonderful!
Monica: By the way, today is a holiday
(shops closed).
Amy: You are telling me now?!
Monica: Let us go to the central park!
```

Such cognitive dissonance as illustrated above frequently

Mehrdad Zakershahrak, Shashank Rao Marpally, Akshay Sharma, Ze Gong, and Yu Zhang are with the School of Computing, Informatics and Decision Systems Engineering, Arizona State University, Tempe, AZ. {mzakersh, smarpall, ashar204, zgong11, yu.Zhang.442}@asu.edu

occurs in our lives. Our goal in this work to avoid similar situations when a robot is making an explanation. The challenge lies in modeling the *changes of context* that are hypothesized to correlate with the cognitive effort for assimilating new information. With such a model, we can determine the order for communicating information that reduces the cognitive effort. To this end, a general formulation based on goal-based Markov Decision Processes is presented. We propose to learn a quantification of the cognitive effort as the reward function in an inverse reinforcement framework [14], [15], [16]. To allow generalization, we use both domain-dependent and domain-independent features to relate to the rewards associated with the changes of context. Learning is based on explanations supported via human subject studies. We set out to test the following hypotheses:

- H1. Our method can learn to predict humans' preferred order for communicating information.

Our results verified H1, suggesting that interpreting an explanation is a progressive process with changes of context.

- H2. Generating explanations based on the learned reward function reduces the cognitive effort.

Our results verified H2, suggesting that order indeed matters for explanation generation!

- H3: Progressive explanations improve task performance.

A comparison with two baseline methods verified H3.

II. Related Work

Explainable AI [17] is increasingly considered an essential paradigm for designing future intelligent agents, especially as such systems begin to constitute a necessary part of our lives. The critical requirement of explainable agency [18] is to be "*explainable*" to the human partners. To be explainable, an agent must provide a solution to achieve a goal and make sure that it is perceived as assistive by its human peers. A determinant here is the human's interpretation of the agent's solution. It is critical to take careful steps to avoid situations where the agent's assistance would be interpreted as no more than an interruption, which leads to the loss of situation awareness and trust [19], [20], and contributes to the pitfalls of earlier effort in designing intelligent assistants.

The key challenge to explainable robotics hence is the ability to model the human's cognitive model that is responsible for interpreting the behaviors of other agents [1]. With such a model, there are different ways to make the robot's behavior explainable. One way is to bias the robot's behavior towards its expectations based on the human's cognitive model. Under this framework, a robot can generate legible motions [21] or explicable plans [22], [23]. Essentially, the robot sacrifices optimality to respect the human's expectation–the resulting plan is often more costly. Another way is to provide forewarning of the robot's intention before execution. In [24], such forewarning is realized by providing additional contexts to help explain the robot's decision. The third way, which is the most relevant to ours, is for the robot to explain its decision via explanations [5], [6], [7]. The advantage of explanation generation, compared to generating explainable

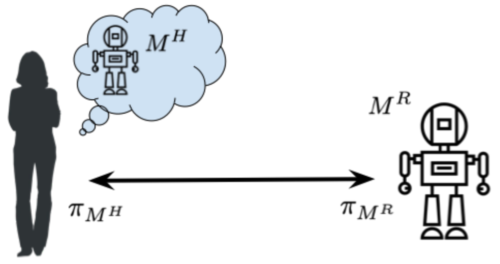

Fig. 1: The model reconciliation setting [9]: M^R denotes the robot's model and M^H denotes the human's model used to generate an expectation of the robot's behavior (i.e., π_{M^H}).

plans, is that the robot can keep its plan optimal. However, as mentioned earlier, the focus in explanation generation has thus far been focused on providing the rationale behind the explainer's decision-making while mostly ignoring the explainee. In [9], this gap is addressed by considering explanation generation as a model reconciliation problem, which considers the explainee's model. The goal there is to reconcile (i.e., reduce) the differences in domain models so that the robot's plan would be interpretable also in the model of the human (i.e., explainee).

The motivation of progressive explanation generation may be compared to that of minimizing effort in replanning [25]. While progressive explanation generation is about reducing the cognitive effort for interpretation, replanning is about reducing the computational costs [26]. However, both involve the processing of additional information given the current context. Our results suggest that there may be a deeper connection between them.

III. Model Reconciliation

We base our work on the model reconciliation setting that considers both the models of the explainer and explainee, which is introduced in [9]. As shown in Fig. 1, the human uses M^H to generate her expectation of the robot's behavior, while the robot's behavior is being created under the robot's model M^R that is different from M^H. Therefore, π_{M^R}, the plan created under M^R, could be different from π_{M^H}, the plan created under M^H, given a planning problem. Whenever these two plans differ, the robot's plan must be explained. Both M^R and M^H are assumed to be known.

Definition 1 (Model Reconciliation [9]): Given a planning problem specified as an initial and goal state pair (I, G), the model reconciliation setting is a tuple $(\pi^*_{I,G}, \langle M^R, M^H \rangle)$ where $cost(\pi^*_{I,G}, M^R) = cost^*_{M^R}(I, G)$: $\pi^*_{I,G}$ is the robot's chosen plan for (I, G) under M^R.

$cost(\pi^*_{I,G}, M^R)$ above returns the cost of the robot's chosen plan under the model M^R; $cost^*_{M^R}(I, G)$ returns the cost of the optimal plan given the initial state and goal state pair under M^R. Therefore, the constraint of $cost(\pi^*_{I,G}, M^R) = cost^*_{M^R}(I, G)$ requires that the robot always chooses an optimal plan. It is also assumed that the human is rational when interpreting the root's plan. It is not difficult to extend the formulation to noisily rational humans.

In such a setting, when $\pi^*_{I,G}$ is not optimal in M^H, the robot must generate an explanation to modify the human's model M^H such that $\pi^*_{I,G}$ becomes optimal (and hence explainable) in the human's modified model (denoted as $\widehat{M^H}$). As a result, an explanation in a model reconciliation setting can be considered as requesting changes to the human's model. Note that making explanations can help with debugging M^R if the robot's model was incorrect. In this paper, however, we consider only that the human model is missing information (as in [9]) that is present in the truth domain model captured by M_R.

To specify model changes, a model function $\Gamma : \mathcal{M} \rightarrow 2^F$ was defined in [9] to convert a model to a set of model features, where $\mathcal{M}$ is the model space and F the feature space. In this way, one model can be updated to another model with editing functions that change one feature at a time. The set of missing features in M_2 with respect to M_1 is denoted as $\Delta(M_1, M_2) = \Gamma(M_1) \setminus \Gamma(M_2)$ and the distance between the two models is the size of $\Delta(M_1, M_2)$, denoted as $\delta(M_1, M_2)$. In this work, we assume that the models are specified as PDDL domain models [27], which are extensions of STRIPS models [28]. A domain model $D = (P, A)$ comprises a set of predicates, P, and a set of actions, A. A state $s \subseteq P$ is a subset of predicates that are true. Each action $a \in A$ can change a state by adding or deleting predicates. An action is represented as a tuple $a = (pre(a), eff^+(a), eff^-(a), c_a)$, where $pre(a)$ denotes the preconditions of the action, and $eff^+(a)$ and $eff^-(a)$ are add and delete effects, respectively. c_a is the cost of the action. For instance, an example of domain model and problem for Amy in our motivating scenario would be:

```
Initial state: not-holiday
Goal state: happy
Actions:
OUTLET-SHOPPING
  pre: not-holiday (car-ready is-sunny)
  eff+: happy
VISIT-PARK
  pre: (car-ready is-sunny)
  eff+: happy
```

For simplicity, we use only predicates without arguments above. Predicates in the parentheses are preconditions that are nice to have but not required. The goal is to achieve the effect of happy. In this example, the domain model, denoted as M_{Amy}, will be converted by the model function Γ to:

```
Γ(M_Amy) = {
     OS-has-precondition-not-holiday,
     OS-has-precondition-car-ready-optional,
     OS-has-precondition-is-sunny-optional,
     OS-has-add-effect-happy, ...}
```

where OS is short for the action OUTLET-SHOPPING above. The function turns a model into a set of features that fully specifies the model. As a result, we can now specify changes to a model by specifying changes to its feature set.

Definition 2 (Explanation Generation [9]): The explanation generation problem in a model reconciliation setting $(\pi^*_{I,G}, \langle M^R, M^H \rangle)$ is the problem to search for an explanation as a subset of $\Delta(M_R, M_H)$ such that:
1) $\Gamma(\widehat{M^H}) \setminus \Gamma(M^H) \subseteq \Gamma(M^R)$, and
2) $cost(\pi^*_{I,G}, \widehat{M^H}) - cost^*_{\widehat{M^H}}(I, G) < cost(\pi^*_{I,G}, M^H) - cost^*_{M^H}(I, G)$.

where $\widehat{M^H}$ denotes the human's model after the changes. The first condition requires the changes to the human's model to be consistent with the robot's model. The second condition states that the robot's chosen plan must be closer (in terms of cost) to the optimal plan after the model changes than before–an explanation should be moving the robot's chosen plan closer to the optimal plan under the human's model.

Definition 3 (Complete Explanation): A complete explanation [9] is an explanation that additionally satisfies $cost(\pi^*_{I,G}, \widehat{M^H}) = cost^*_{\widehat{M^H}}(I, G)$.

A complete explanation requires the changes to make the robot's chosen plan optimal in the updated human model.

IV. Progressive Explanation Generation

We can now formulate the problem of progressive explanation generation. Recall that the problem here is to search for an order for communicating information in an explanation that reduces the cognitive effort of interpretation. Since an explanation in the model reconciliation setting is a set of feature changes, we associate an interpretation cost with each unit feature change. The cognitive effort required for interpreting an explanation hence becomes the cumulative cost at each step. Hence, we model the interpretation cost for each change with a *model distance metric*, denoted as $\rho(M_i, M_{i+1})$ for the ith feature change, where M_i is the model before the i-th feature change and M_{i+1} is the model after that change. Thereby, progressive explanation generation can be defined as an optimization problem:

Definition 4 (Progressive Explanation Generation (PEG)): A progressive explanation is a complete explanation with an ordered set of unit feature changes that minimize the sum of the model distance metric: $\arg\min_{\Delta(\widehat{M^H}, M^H)} \sum_{f_i \in \langle \Delta(\widehat{M^H}, M^H) \rangle} \rho_i$, where ρ_i is short for $\rho(M_i, M_{i+1})$, i is the index of unit feature changes, and f_i denotes the i-th unit feature change.

The angle brackets above denote an ordered set. The challenge remains is the computation of the model distance metric. This metric depends on the context when an unit feature change is made (i.e., communicated to the human) and cannot be directly computed. We use a learning approach.

A. Learning the Model Distance Metric

To learn the model distance metric for PEG, we formulate the problem as an inverse reinforcement learning (IRL) [14], [15], [16] framework, where we assume the task of generating explanations can be expressed as a goal-based Markov Decision Processes (MDP). A goal-based MDP is defined by a 6-element tuple (S, A, T, R, γ, G), where S is the state space and A is the action space. The domain dynamics is represented as the transition function T that determines the probability of transitioning into state s' when taking an action a in state s (i.e., $P(s'|s, a)$). R is the reward function, and the goal of the agent is to maximize the expected cumulative reward. γ is the discount factor that encodes the agent's preference of short-term rewards

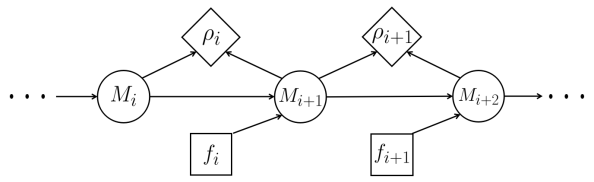

Fig. 2: Illustration of the MDP that underlies PEG. At each time step, the human's model M_i serves as the state.

over long-term rewards. G is a set of goal states where for each $g \in G$, $T(g, a, g) = 1, \forall a \in A$. We chose goal-based MDPs since explanation generation in a given domain could have different explanation goals depending on the planning problem (I, G) in the model reconciliation scenario considered: the feature set to be included in the explanation and communicated may differ from scenario from scenario.

Fig. 2 demonstrates the MDP that underlies PEG. In our work, the state space S is the set of all possible human models, and the action space A is the set of all possible unit feature changes. The transition function T captures the probability that the human model would be updated to M' when the human model is M and the robot explains f to her (i.e., $P(M'|M, f)$). The model distance metric ρ serves as the reward function, which depends on both the current and next human models.

B. Applying IRL

Following prior work on IRL [14], [15], [16] and to allow generalization, we model the distance metric as a linear combination of a set of weighted features:

$$\rho(M, M') = \sum_i \theta_i \cdot \psi_i(M, M') = \Theta^T \Psi(M, M')$$

where $\Psi = \{\psi_1, \psi_2, \ldots, \psi_k\}$ is the set of features with respect to the state pair (M, M'). $\Theta = \{\theta_1, \theta_2, \ldots, \theta_k\}$ is the set of weights for the features.

Given a set of traces as a set of explanations (each is a sequence of unit features changes under different model reconciliation scenarios) obtained from human subjects, our goal is to learn the model distance metric ρ, which in turn requires us to learn the weights Θ given a set of features. Since noise is expected in the traces, we learn the weights by maximizing the likelihood of the traces using MaxEnt-IRL [16] as follows:

$$\begin{aligned}\Theta^* &= \arg\max_\Theta \mathcal{L}(D) = \arg\max_\Theta \frac{1}{|D|} \log P(D|\Theta) \\ &= \arg\max_\Theta \frac{1}{|D|} \sum_{G \in \mathcal{G}} \sum_{\widehat{\zeta_G} \in D_G} \log P(\widehat{\zeta_G}|\Theta)\end{aligned} \tag{1}$$

where D is the training data set, and $\mathcal{G}$ the collection of different scenarios. To simplify the discussion, we keep the initial state I fixed, and thus the different scenarios can be indexed by the goal state G. Hence, $\widehat{\zeta_G} = (M_0, f_1, M_1, \ldots, f_n, M_n)$ is an explanation for the scenario (I, G) with a sequence of feature changes provided by human subjects from the set D_G. It consists of the initial human model (i.e., $M_0 = M^H$), unit feature change and the updated model at each time step. To mitigate the ambiguity that the distribution of the traces may introduce preference for some traces over others, the principle of maximum entropy [16] is employed for the distribution over all the possible traces for a specific goal G:

$$P(\zeta_G|\Theta) = \frac{e^{\rho(\zeta_G)}}{\sum_{\zeta_G} e^{\rho(\zeta_G)}} \tag{2}$$

where

$$\rho(\zeta_G) = \Theta^T \Psi(\zeta_G) = \sum_{(M,M') \in \zeta_G} \Theta^T \Psi(M, M')$$

Take Equation 2 into Equation 1, the optimization becomes:

$$\Theta^* = \arg\max_\Theta \frac{1}{|D|} \sum_{G \in \mathcal{G}} \sum_{\widehat{\zeta_G} \in D_G} \Big(\Theta^T \Psi(\widehat{\zeta_G}) - \log \sum_{\zeta_G} e^{\Theta^T \Psi(\zeta_G)}\Big) \tag{3}$$

Note that $\widehat{\zeta_G} \in D_G$ in the first term above represents a trace in the training set while ζ_G in the second term refers to any possible trace of the domain. Since Equation 3 is convex, we apply a gradient-based method to learn Θ and divide the traces into transitions as in [16]:

$$\nabla_\Theta \mathcal{L} = \frac{1}{|D|} \sum_{G \in \mathcal{G}} \Big(\sum_{(M,M') \in D_G} \Psi(M, M') - \sum_{(M,M') \in D_G} P(M, M'|\Theta) \Psi(M, M') \Big)$$

Different from the traditional applications of MaxEnt-IRL [16], the model distance metric in our work depends on both the current and next human model. As a result, $P(M, M'|\Theta)$ above represents the model pair occurrence frequency (MPOF) for a pair (M, M'), which can be computed using dynamic programming. If we denote the probability of occurrence of (M, M') at time t as $\mu_t(M, M')$, we then have $P(M, M'|\Theta) = \sum_t \mu_t(M, M')$. The updating rules for μ_t is as follows:

$$\begin{aligned}\mu_0(M, M') &= P\Big((M_0, M_1) = (M, M')\Big) \\ \mu_{t+1}(M, M') &= \sum_f \sum_{M''} \mu_t(M'', M) P(f|M) P(M'|M, f)\end{aligned}$$

The values for μ_0 are initialized to the probability of the state pair (M, M') being the first pair of a trace. The probability of the occurrence of (M, M') at a certain time step then is calculated based on the occurrence frequency of the previous state pair, which has M as the second entry in the pair, any unit feature change f that the robot would explain to the human while in state M (i.e., according to a stochastic policy), and the probability that the human model would end up in M' when explaining f in state M according to the transition function.

The stochastic policy $P(f|M)$ specifies the probability of explaining f when the human model is M, which is computed as $P(f|M) = \frac{P(M,f)}{P(M)}$. Similarly, they can be calculated using dynamic programming as in [16]. $\mu_t(M, M')$ can then be approximated using sampled traces generated by the stochastic policy and transition function in each iteration. After learning the parameters for the model distance metric, we utilize a uniform cost search for a given goal G to retrieve the best sequence of f_i from the initial state by maximizing the accumulative reward:

$$\zeta_G^* = \arg\max_{\zeta_G} \sum_{(M,M')\in\zeta_G} \Theta^T \Psi(M, M') \tag{4}$$

C. Features Selection

We consider both domain dependent and independent features for learning the model distance metric. Features that best inform the cognitive effort for interpreting new information should be selected. In explanation generation, these features are specified for each transition, which is associated with the change of context as a result of unit feature changes. Hence, each distinct unit feature change is naturally a domain dependent feature. However, we found them to play a less prominent role compared to more informative ones used in our evaluation.

Domain independent features are more difficult to select. Given our focus on planning tasks, we anticipate that the change of context is tied to the change of plan as new information is received. The cognitive effort required in such cases reflects the difficulty of discovering a new plan based on the current plan. Hence, we use various plan distances as domain independent features, which capture how any two plans differ. In particular, we consider three plan distance metrics: (1) action distance [25], (2) cost distance, and (3) Levenshtein distance [29] (i.e., minimum editing distance). These distances are discussed more formally below. First, we are given the model reconciliation scenario as a planning problem specified by (I, G). For any model M_i, we denote the optimal plan for (I, G) as π_i. To simplify the discussion, we assume there is a single optimal plan only.

Action Distance: The action distance feature represents the distance between two plans π_i and π_j obtained under the models M_i and M_j, respectively, or written as $d_{action}(\pi_i, \pi_j) = \frac{\sum_{k=1}^{n} |N_i(a_k) - N_j(a_k)|}{max(len(\pi_i), len(\pi_j))}$ where n is the number of distinct actions in π_i and π_j considered together, $N_i(a_k)$ is the number of occurrences of action a_k in plan π_i, and $len(\pi_i)$ is the length of plan π_i.

Cost Distance: The cost distance is the difference between the cost of plans π_i and π_j obtained under M_i and M_j, respectively: $d_{cost}(\pi_i, \pi_j) = |cost^*_{M_i}(I, G) - cost^*_{M_j}(I, G)|$.

Levenshtein Distance: The Levenshtein distance [29] is the minimum editing distance between plans π_i and π_j, defined recursively as follows:

$$d_{lav}(m,n) = \begin{cases} \max(m,n) & \text{if } \min(m,n) = 0 \\ \min \begin{cases} d_{lav}(m-1,n)+1 \\ d_{lav}(m,n-1)+1 \\ d_{lav}(m-1,n-1) + \mathbf{1}_{\pi_i[m]\neq\pi_j[n]} \end{cases} & \text{otherwise.} \end{cases}$$

where $d_{lav}(m,n)$ is the Levenshtein distance between the first m actions in π_i and first n actions in π_j. $\mathbf{1}_{\pi_i[m]\neq\pi_j[n]}$ returns 0 when $\pi_i[m] = \pi_j[n]$ and 1 otherwise.

V. Evaluation

We evaluated our approach by conducting human subject studies using Amazon Mechanical Turk (MTurk) in a synthetic domain called escape-room. This domain is designed to expose the subjects to moderately complex planning tasks that require a non-trivial amount of cognitive effort in a short amount of time. To improve the responses' quality, we set the criteria that the worker's HIT acceptance rate must be greater than 99% and has been granted MTurk Masters.

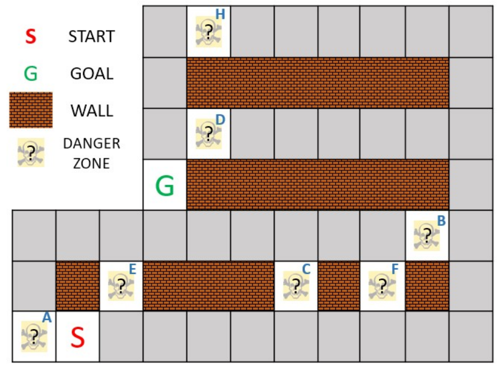

Fig. 3: Illustration of the escape-room domain.

A. Escape Room

The task is situated in a damaged nuclear plant represented as a maze-like environment in Fig. 3. The goal of the internal agent is to navigate from the starting location S to the goal location G to escape as fast as possible without going through dangerous locations. The set of actions in this domain includes going to each of the gateway cells (i.e., marked by an English letter on the top right) from S, and then to G. Some of the locations with a question mark (see Fig. 3) may be affected by the disaster and become dangerous. There is also an external agent who knows the locations that are affected and can communicate with the internal agent who does not have this information initially. The external agent can only convey one piece of information at a time (e.g., D is a danger zone). The states of the 7 locations with a question mark correspond to 7 contingencies (modeled as unit feature changes) that can affect the internal agent's plan. For domain dependent features, we chose 4 features related to the maximum (and minimum) relative x and y positions of the contingency being explained with respect to the contingencies that have already been explained. We refer to these features as x_{min}, x_{max}, y_{max}, and y_{min}.

1) Experimental Design: We designed 8 different scenarios for the escape-room domain. We used 5 scenarios for training and 3 for testing. Each scenario involves a different set of contingencies, and we ensure that there are contingencies in the testing scenarios that did not appear in the training scenarios. During training, the participants were at first introduced to the domain and informed that they must act as the external agent to communicate the contingencies to the internal agent. They were explained that the internal agent was desperate to escape to give them a sense of urgency and incentivize them to elucidate the situation. We also asked

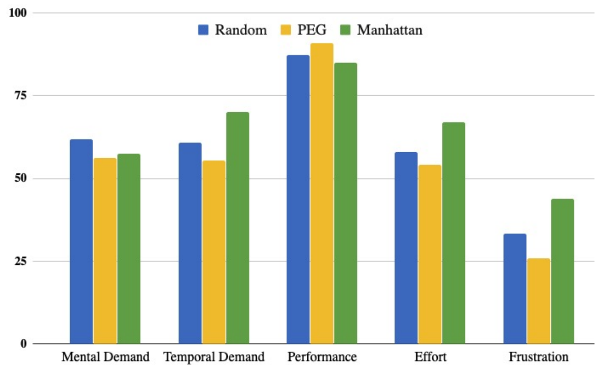

Fig. 4: NASA TLX results for testing scenarios

the participants at the beginning about what path the internal person would take, assuming all the locations marked were safe. We used the answer to sift the data.

For testing, new participants were invited to play the role of the internal agent now with automated explanations generated by different algorithms. We compared the participants' performance with our progressive explanation generation method and two baselines, which included one with random order, and one that used only domain dependent features to determine the order using our approach. The participants were asked to complete the task within 4 minutes. Responses that failed the sanity check question or ran over time were not used. After the task, the participants were provided the NASA Task Load standard questionnaire (TLX) [30].

2) Results & Analysis: We created the surveys using Qualtrics. In the training phase, we recruited 35 participants on MTurk, out of which 21 responses were used. For testing, we recruited 163 participants, out of which 87 responses were used. 58 of our participants were male, and 29 were female. The average age of our subjects was 38.17 with a standard deviation of 11.13.

Table I shows the normalized weights Θ for each feature after IRL as explained in Sec. IV-B. Interestingly, the action distance and Levenshtein distance maintained the highest weights. This result aligned with our expectation that the interpretation of an explanation was a progressive process and context dependent. On the other hand, the weight for the cost distance was the lowest, which revealed that cost played a less important role in determining the cognitive effort.

Feature Category	Feature Name	Weights
Domain independent	Action Distance	0.44
	Cost Distance	0.04
	Levenshtein Distance	0.46
Domain dependent	x_{min}, y_{min}	0.38, 0.41
	x_{max}, y_{max}	0.35, 0.39

TABLE I: Normalized feature weights

The subjective results for testing scenarios are presented in Fig. 4. We can see that our method (PEG) performed better than the baselines in all NASA TLX metrics. A statistically significant difference was observed between PEG and other methods for the (equally) weighted sum of TLX metrics, as shown in Table II. It is clear that our method can generate more user-friendly explanations (H1) that reduce the cognitive effort (H2). Objective metrics further confirmed that our method improved task performance. In particular, the percentage in which the participants came up with the correct plan that was safe for the respective methods: : 85.4% in random, 96.3% in PEG, and 66.7% in Manhattan (H3).

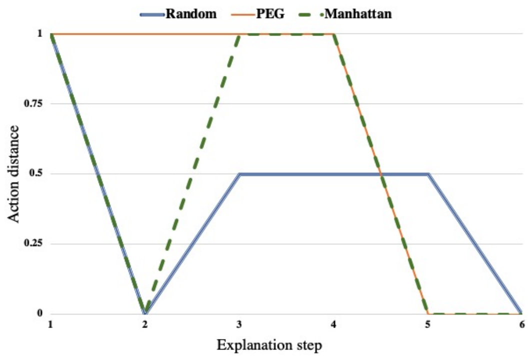

Fig. 5: Changes of action distance per each unit change

	Mental Demand	Temporal Demand	Perform.	Effort	Frustration	Weighted Sum (excluding Perf.)
Random	63.10	61.96	89.06	59.04	33.96	52.47
PEG	**56.19**	**55.37**	**90.74**	**54.11**	**25.89**	**41.93**
Manhattan	57.43	69.93	85.00	66.86	43.93	58.35

TABLE II: Subjective results for each NASA TLX category

Fig. 5 shows the action distance between the current plan to the final plan per each unit feature change in one of the testing scenarios. An interesting observation is that the curve of PEG seemed smoother, i.e., it appeared to prefer context changes in one direction (see our motivating scenario for an example where the context changes in different directions). This intuitively makes sense since context change in different directions only increases the extent of changes needed later.

VI. Conclusions

In this paper, we studied the problem of PEG. We took a step further from the prior work by considering not only the right explanation for the explainee, but also the underlying cognitive effort required for understanding the explanation. To learn the order for communicating information that assists understanding, we adopted a goal-based MDP and applied IRL to learn the reward function based on traces, which is then used to generate explanations. We showed that order for communicating information indeed mattered! First, we noted strong weights for domain independent features, suggesting that the cognitive effort for interpreting an explanation was tightly coupled with the context changes as new information was received progressively. Finally, we showed that PEG did improve task performance and reduce cognitive load.

Acknowledgement

We thank the anonymous reviewers for their helpful comments. This research is supported in part by the NSF grant IIS-1844524, the NASA grant NNX17AD06G, and the AFOSR grant FA9550-18- 1-0067.

References

[1] T. Chakraborti, S.Kambhampati, M.Scheutz, and Y. Zhang, “Ai challenges in human-robot cognitive teaming,” 2017.

[2] N. J. Cooke, “Team cognition as interaction,” *Current directions in psychological science*, vol. 24, no. 6, pp. 415–419, 2015.

[3] T. Lombrozo, “The structure and function of explanations,” *Trends in cognitive sciences*, vol. 10, no. 10, pp. 464–470, 2006.

[4] M. R. Endsley, “Design and evaluation for situation awareness enhancement,” in *Proceedings of the Human Factors Society annual meeting*, vol. 32. SAGE Publications Sage CA: Los Angeles, CA, 1988, pp. 97–101.

[5] M. Göbelbecker, T. Keller, P. Eyerich, M. Brenner, and B. Nebel, “Coming up with good excuses: What to do when no plan can be found,” in *ICAPS*, 2010.

[6] M. Hanheide, M. Göbelbecker, G. S. Horn, A. Pronobis, K. Sjöö, A. Aydemir, P. Jensfelt, C. Gretton, R. Dearden, M. Janicek, *et al.*, “Robot task planning and explanation in open and uncertain worlds,” *AIJ*, vol. 247, pp. 119–150, 2017.

[7] S. Sohrabi, J. A. Baier, and S. A. McIlraith, “Preferred explanations: Theory and generation via planning,” in *AAAI*, 2011.

[8] T. Chakraborti, A. Kulkarni, S. Sreedharan, D. E. Smith, and S. Kambhampati, “Explicability? legibility? predictability? transparency? privacy? security? the emerging landscape of interpretable agent behavior,” in *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling, ICAPS 2018, Berkeley, CA, USA, July 11-15, 2019*, J. Benton, N. Lipovetzky, E. Onaindia, D. E. Smith, and S. Srivastava, Eds. AAAI Press, 2019, pp. 86–96. [Online]. Available: https://aaai.org/ojs/index.php/ICAPS/article/view/3463

[9] T. Chakraborti, S. Sreedharan, Y. Zhang, and S. Kambhampati, “Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. ijcai (2017), 156–163,” 2017.

[10] M. Zakershahrak, Z. Gong, N. Sadassivam, and Y. Zhang, “Online explanation generation for planning tasks in human-robot teaming,” in *International Conference on Intelligent Robots and Systems*, 2020.

[11] K. A. Ericsson and J. Smith, *Toward a general theory of expertise: Prospects and limits*. Cambridge University Press, 1991.

[12] D. Kahneman, *Thinking, fast and slow*. Macmillan, 2011.

[13] C. V. Schwarz, B. J. Reiser, E. A. Davis, L. Kenyon, A. Achér, D. Fortus, Y. Shwartz, B. Hug, and J. Krajcik, “Developing a learning progression for scientific modeling: Making scientific modeling accessible and meaningful for learners,” *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching*, vol. 46, no. 6, pp. 632–654, 2009.

[14] A. Y. Ng and S. J. Russell, “Algorithms for inverse reinforcement learning,” in *ICML*. Morgan Kaufmann Publishers Inc., 2000, pp. 663–670.

[15] P. Abbeel and A. Y. Ng, “Apprenticeship learning via inverse reinforcement learning,” in *Proceedings of the twenty-first ICML*. ACM, 2004, p. 1.

[16] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning,” in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, ser. AAAI08. AAAI Press, 2008, p. 14331438.

[17] D. Gunning, “Explainable artificial intelligence (xai),” *Defense Advanced Research Projects Agency (DARPA), nd Web*, vol. 2, 2017.

[18] P. Langley, B. Meadows, M. Sridharan, and D. Choi, “Explainable agency for intelligent autonomous systems,” in *IAAI*, 2017.

[19] M. R. Endsley, *Designing for situation awareness: An approach to user-centered design*. CRC press, 2016.

[20] C. W. Langfred, “Too much of a good thing? negative effects of high trust and individual autonomy in self-managing teams,” *Academy of management journal*, vol. 47, no. 3, pp. 385–399, 2004.

[21] A. D. Dragan and S. S. Srinivasa, “Generating legible motion,” 2013. [Online]. Available: http://www.roboticsproceedings.org/rss09/p24.html

[22] Y. Zhang, S. Sreedharan, A. Kulkarni, T. Chakraborti, H. H. Zhuo, and S. Kambhampati, “Plan explicability and predictability for robot task planning,” in *ICRA*. IEEE, 2017.

[23] M. Zakershahrak, A. Sonawane, Z. Gong, and Y. Zhang, “Interactive plan explicability in human-robot teaming,” in *RO-MAN)*. IEEE, 2018, pp. 1012–1017.

[24] Z. Gong and Y. Zhang, “Behavior explanation as intention signaling in human-robot teaming,” in *RO-MAN*. IEEE, 2018, pp. 1005–1011.

[25] M. Fox, A. Gerevini, D. Long, and I. Serina, “Plan stability: Replanning versus plan repair.” in *ICAPS*, vol. 6, 2006, pp. 212–221.

[26] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, “Anytime dynamic a*: An anytime, replanning algorithm.” in *ICAPS*, vol. 5, 2005, pp. 262–271.

[27] M. Fox and D. Long, “Pddl2. 1: An extension to pddl for expressing temporal planning domains,” *JAIR*, vol. 20, pp. 61–124, 2003.

[28] R. E. Fikes and N. J. Nilsson, “Strips: A new approach to the application of theorem proving to problem solving,” *AIJ*, vol. 2, no. 3-4, pp. 189–208, 1971.

[29] V. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” in *Soviet physics doklady*, vol. 10, no. 8, 1966, pp. 707–710.

[30] NASA, “Nasa task load index,” https://humansystems.arc.nasa.gov/groups/TLX/, Jan. 2020.