Iterative Planning with Plan-Space Explanations: A Tool and User Study

Rebecca Eifler and **Jörg Hoffmann**
Saarland University, Saarland Informatics Campus, Germany
{eifler, hoffmann}@cs.uni-saarland.de

Abstract

In a variety of application settings, the user preference for a planning task – the precise optimization objective – is difficult to elicit. One possible remedy is planning as an iterative process, allowing the user to iteratively refine and modify example plans. A key step to support such a process are explanations, answering user questions about the current plan. In particular, a relevant kind of question is "Why does the plan you suggest not satisfy p?", where p is a *plan property* desirable to the user. Note that such a question pertains to plan space, i. e., the set of possible alternative plans. Adopting the recent approach to answer such questions in terms of *plan-property dependencies*, here we implement a tool and user interface for human-guided iterative planning including plan-space explanations. The tool runs in standard Web browsers, and provides simple user interfaces for both developers and users. We conduct a first user study, whose outcome indicates the usefulness of plan-property dependency explanations in iterative planning.

1 Introduction

In many real life settings, like space mission control, production planning in Industry 4.0, or robot-aided disaster recovery, typically not all goals, constraints and preferences are known from the beginning. There may even be different user groups with different preferences. Take for instance researchers from different fields in a mission control center, who all have to be satisfied with the plan. Given this setting, the traditional planning workflow – select a set of goals, compute a plan, execute – is not adequate. Instead planning should support the users in making up their minds, exploring plan space until they are satisfied.

A natural framework for the latter is *planning as an iterative process* (Smith 2012), as illustrated in Figure 1. This allows the human users to iteratively refine their preferences and goals based on example plans. The possibility to provide explanations, answering user questions about the current plan, is a key feature in this setting. In particular, user questions of the form "Why does the suggested plan π not satisfy preference p?" are relevant. Such explanations allow the user to develop a deeper understanding of the space of plans. This in turn enables the user to refine his goals and preferences accordingly.

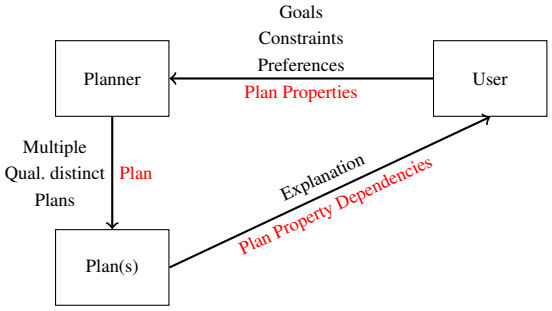

Figure 1: Planning viewed as an iterative process of plan revision under constraints. black: generic description by Smith (2012); red: our instantiation.

Eifler et al. (2020a; 2020b), henceforth referred to as *Eif20*, introduced a framework addressing the problem of generating explanations via **plan-property entailments**. A plan property p is a Boolean function on plans, and p entails q if all plans that satisfy p also satisfy q. The above stated question could then be answered with "Because if you achieve p you have to forego p'" . This is a form of contrastive explanation (Miller 2019). Some other works (Smith 2012; Fox, Long, and Magazzeni 2017; Cashmore et al. 2019; Krarup et al. 2019) propose to answer such questions by highlighting the differences between a plan π not satisfying p and an alternative plan π' that does. Observe that both approaches naturally shed light on the trade-off between different user preferences.

Eif20 assume that the set P of plan properties is given as an input. They then address oversubscription planning (OSP) (Smith 2004; Domshlak and Mirkis 2015) where not all of P can be satisfied. Based on **exclusion dependencies** of the form $\bigwedge_{p\in X} p \Rightarrow \neg \bigwedge_{p\in Y} p$ where $X, Y \subseteq P$, they are able to answer the type of questions stated above. They also propose an *online* version which allows to adapt the set P during planning. Arguably, this framework perfectly fits the iterative planning workflow described by Smith (2012). The goals, constraints and preferences can be expressed as plan properties. Given this representation, the explanations are then naturally given as plan property dependencies.

Here we contribute a Web-based iterative planning tool, realizing the workflow suggested by (Smith 2012) and instantiating the explanation part with Eif20's framework. The tool enables users to perform iterative planning with plan properties representing goals and preferences. One can enforce selected plan properties reflecting changing preferences across planning iterations, one can ask questions about the iteratively refined plans, and one can add new plan properties to hone in on new issues that become apparent during the iterative planning process. To accommodate layperson users unfamiliar with planning, our tool comprises a simplified version, with a fixed set of plan properties, and with an enriched visualization of the planning task. The tool runs in standard Web browsers, and provides simple user interfaces for both, case-study developers and users.

We furthermore conduct a first user study evaluating our tool and therewith providing the first user-based evaluation of Eif20's framework. Specifically, we evaluate the usefulness of Eif20's explanation facilities, by comparing the use of our tool with vs. without these facilities. While the user study is small and preliminary, its outcome is encouraging, indicating the usefulness of plan-property dependency explanations in iterative planning.

Section 2 gives some background on the planning formalism and Eif20's framework. Section 3 describes the tool implementation. The preliminary user study is described and evaluated in Section 4. Section 5 lists work we want to address in the future and concludes the paper.

2 Background

Oversubscription Planning

Our framework is based on a finite-domain variable variant of oversubscription planning (OSP) (Smith 2004; Domshlak and Mirkis 2015). An **OSP task** is a tuple $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$ where V is the set of **variables**, A is the set of **actions**, $c : A \to \mathbb{R}_0^+$ is the action **cost** function, I is the **initial state**, G^{hard} (G^{soft}) is the **hard** (**soft**) **goal**, and $b \in \mathbb{R}_0^+$ is the **cost bound**. A **state** is a complete assignment to V; G^{hard} and G^{soft} are partial assignments to V, defined on disjoint sets of variables; each action $a \in A$ has a **precondition** pre_a and an **effect** eff_a, both partial assignments to V. We refer to variable-value pairs $v = d$ as **facts**, and we identify partial variable assignments with sets of facts. An action a is **applicable** in a state s if $pre_a \subseteq s$. The outcome state $s[[a]]$ is the same as s except that $s[[a]](v) = eff_a(v)$ for those v on which eff_a is defined. The outcome state of an iteratively applicable action sequence π is denoted by $s[[\pi]]$. A **plan** is an action sequence π whose summed-up cost is $\leq b$ and where $G^{\text{hard}} \subseteq I[[\pi]]$.

We follow Eif20 in not defining a plan utility over G^{soft}. Instead, G^{soft} is a set of plan properties – including more general plan properties compiled into goal facts – and the analysis we provide identifies dependencies between these plan properties. The underlying assumption is that the user's preferences over G^{soft} are difficult to elicitate, and not expressable in a utility function.

As our running example, consider the OSP task illustrated in Figure 2, based on the IPC NoMystery domain where packages must be delivered respecting limited fuel. The figure depicts the initial locations of trucks and packages. Both trucks have initially 3 fuel units and each road section consumes 1 unit of fuel. This is also the example we use in our preliminary user study.

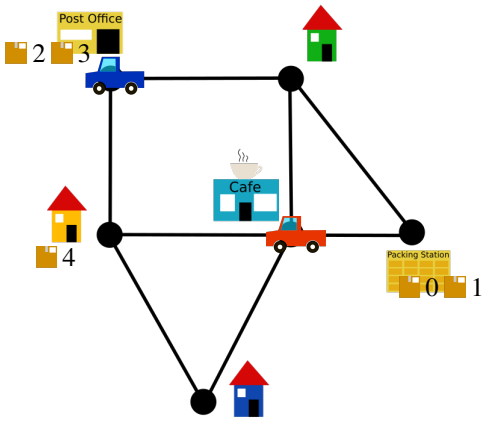

Figure 2: An illustrative NoMystery example.

Eifler et al.'s Framework

Eif20 introduce a framework for plan-property dependency analysis in OSP tasks, and algorithms for a specific instantiation of that framework. For the work presented in the following, we only consider that instantiation, simplifying Eif20's concepts accordingly. In the next section, we outline the definitions that build the foundation of our contribution.

Assume an OSP task $\tau = (V, A, c, I, G^{\text{hard}}, G^{\text{soft}}, b)$. In general, a **plan property** is a function p mapping action sequences π in τ to Boolean values. Eif20's implementation only, considers plan properties p that can be compiled into goal facts. A plan π satisfies a property p iff the corresponding goal fact g_p is true in $I[[\pi]]$. They are interested in conjunctions $\bigwedge_{g \in X} g$ of such plan properties, and specifically in **exclusion dependencies** where $\bigwedge_{g \in X} g$ "entails" $\neg \bigwedge_{g \in Y} g$ $(X, Y \subset P)$.

Entailment here is intended as entailment *in the space of plans*. It is often the case that all *plans* which satisfy X cannot satisfy Y. We denote by Π the set of plans for τ. We say that $\pi \in \Pi$ **satisfies** a formula ϕ over G^{soft}, written $\pi \models \phi$, if ϕ evaluates to true under the truth value assignment where $g \in G^{\text{soft}}$ is *true* iff $g \in I[[\pi]]$. We denote by $\mathcal{M}_\Pi(\phi) := \{\pi \mid \pi \in \Pi, \pi \models \phi\}$ the subset of plans that satisfy ϕ. We say that ϕ Π**-entails** ψ if $\mathcal{M}_\Pi(\phi) \subseteq \mathcal{M}_\Pi(\psi)$.

Applied to exclusion dependencies, this means that $\bigwedge_{g \in X} g$ Π-entails $\neg \bigwedge_{g \in Y} g$ if all action sequences in τ whose cost is within the bound b, that achieve G^{hard}, and that achieve all $g \in X$, do not achieve at least one $g \in Y$.

The problem now is to compute all exclusion dependencies over G^{soft}. This corresponds to preparing answers to all questions of the form "Why does the plan not satisfy the properties in X?".

To this end, observe that $\bigwedge_{g \in X} g$ Π-entails $\neg \bigwedge_{g \in Y} g$ iff $\bigwedge_{g \in X \cup Y} g$ is unsolvable in τ. Observe further that, in this case, the same is true for every $X' \supseteq X$ and $Y' \supseteq Y$, i. e., the exclusion dependency is strongest for set-inclusion minimal X and Y. Given these observations, the problem boils down to computing all **minimal unsolvable goal subsets (MUGS)**: all sets $G \subseteq G^{\text{soft}}$ where G cannot be achieved but every $G' \subsetneq G$ can.

For our example we consider the following plan properties:

1. Package 0 is delivered to the blue house.
2. Package 1 is delivered to the green house.
3. Package 2 is delivered to the blue house.
4. Package 3 is delivered to the orange house.
5. Package 4 is delivered to the post office.
6. Package 4 is delivered to the packing station.
7. The road between the cafe and the packing station is not used by the red truck.
8. The same truck is used for package 0 and 2.
9. The same truck is used for package 2 and 3.
10. The blue truck visits the cafe.
11. Package 0 is delivered before package 1.

The first six of these are standard goal facts; 7 to 11 can be expressed in LTL, which can be compiled into goal facts using well-known techniques (Baier and McIlraith 2006; Edelkamp 2006; Eifler et al. 2020b); 7 to 10 can also be expressed in a simple LTL fragment named *action set properties* by Eif20, for which a very compact compilation into goal facts is possible (and that Eif20 use in their implementation whenever possible).

The MUGS in this example are $\{2, 4, 9\}, \{2, 6\}, \{3, 6, 9\}, \{3, 8\}, \{4, 6, 9, 10\}, \{4, 7, 9\}, \{5, 6\}, \{5, 7\}, \{6, 8, 10\}, \{7, 8\}$. So, for example, the answer to the user question "Why don't you avoid driving from the packing station to the cafe?", could be answered using MUGS $\{7, 8\}$ "Because then you can not deliver package 0 and 2 with the same truck.".

3 Tool Description

We implemented a Web based iterative planning tool. As a planner we use Fast Downward (Helmert 2006) and as an explanation generator the implementation of Eif20. The translation of plan properties to goal facts is realized using Eif20's definitions and implementations for LTL and action-set plan properties.

Users The tool is designed for two different user groups. Technically versed users, familiar with computer science or AI planning; and layperson users, with no computer science background. The adaptations for layperson users focus mainly on the visualization of the planning task and plan properties. The part of the tool dedicated to layperson users is mainly designed to give the possibility to conduct user studies.

Overview

First, we want to give a general overview of the tool structure and the provided functionalities. The tool has three main components, *projects*, *demos* and *user studies*, as listed in the user's home page in Figure 3.

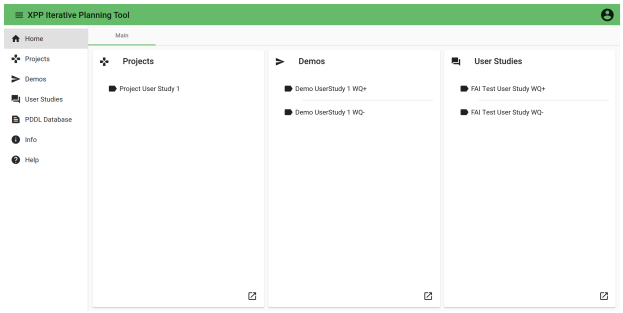

Figure 3: Tool overview: Home page.

A project consists of a planning task definition and plan properties reflecting the goals and user preferences. It provides the full functionality for iterative planning with online computed explanations.

This includes:

- enforcing selected plan properties reflecting changing preferences across planning iterations
- asking questions about the iteratively refined plans
- adding new plan properties to hone in on new issues that become apparent during the iterative planning process

A demo provides iterative planning with a fixed set of plan properties as a simplified version to accommodate layperson users unfamiliar with planning.

The tool also supports the conduction of user studies. A user study can be composed of multiple demos, links to external questionnaires and additional descriptions and instructions.

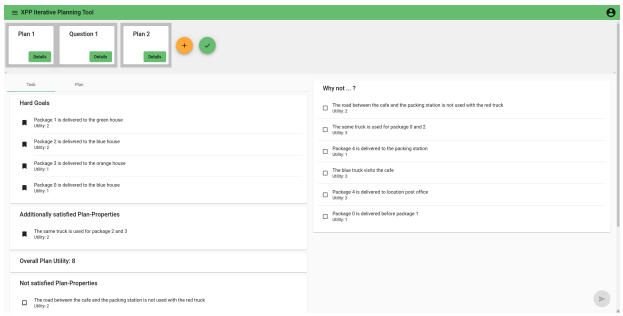

Figure 4: Tool overview: Iterative planning.

An overview of the iterative planning interface is given by Figure 4. At the top, the interface has a navigation bar for plans and questions, which have been computed or asked in previous iterations. On the left there is the hard and soft goal information for the currently selected plan. The interface for posing questions, concerning the current plan, is located on

the right. Both interfaces will be explained in more detail in the next section.

Iterative Planning

In the following, both the input for an iterative planning project, and the interfaces to compute plans and ask question are described in more detail.

Input The task definition is given by a standard PDDL (McDermott et al. 1998) domain and problem file. The problem file only defines the initial state. The definition of goals is handled separately.

The plan properties used during planning have to be defined by an user. They can defined plan properties by explicitly stating the corresponding LTL formula and actions sets. In addition there exits the possibility to define domain dependent templates for plan properties. These map a predefined natural language representation to the corresponding formula. For example, the interface for the plan property template "The road between L_i and L_j is used with truck T_i" is given in Figure 5. For every variable the type can be specified. Furthermore, it is possible to impose constraints between the variables, like L_i and L_j have to be connected. This allows to define plan properties more easily as one only has to select the concrete instantiation.

Figure 5: Interface plan property creation.

As it does not matter to the user how the plan properties are represented internally, a differentiation between the different definition languages is not necessary in the interface. The same holds for plan properties and goal facts from the original planning task, like delivering the packages in our example. Goal facts can also be represented as plan properties, which is the way they are handled in the implementation. This also allows to define different delivery options like the packing station or the post office for package 4 in our example.

Initially, plan properties representing the already known goals and preferences have to be specified by the user. As these might change or new preferences might form during planning, it is possible to add more plan properties reflecting these changes during planning.

A plan property can be declared as a global hard goal. Such plan properties are preselected as hard goals for every planning iteration. In our example the delivery of package 0 is a global hard goal representing a priority package.

Plan Step To compute a plan, the user has to select a subset of the given plan properties. The global hard goals are automatically added. The corresponding interface for the example task is shown in Figure 6. At the top the global hard goals are listed. The plan properties which can be selected by the user as additional hard goals are enumerated below. After confirming the choice, a plan for the selected hard goals is computed.

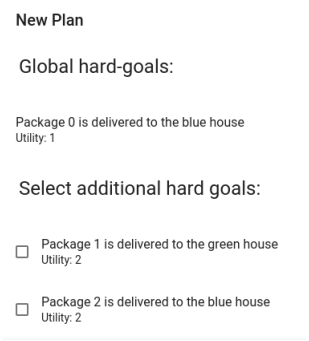

Figure 6: Interface to select plan properties as hard goals.

For a solvable selection, the tool provides the following information:

- the plan properties selected as hard goals
- the plan properties which are additionally satisfied although not enforced
- the unsatisfied plan properties
- the plan as a sequence of actions
- the plan as an animation

Figure 7 depicts the representation of a plan for which no other properties than the global hard goals have been selected. In this instance 2 additional plan properties are satisfied although not enforced.

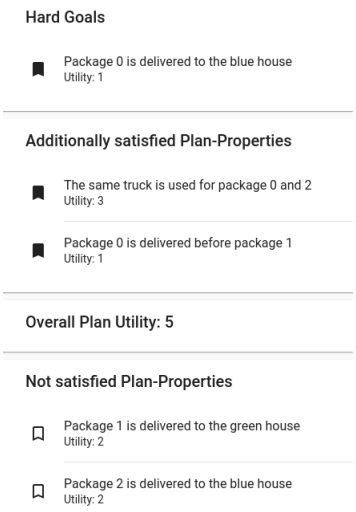

Figure 7: Information about the currently selected plan.

If the task is unsolvable only the hard goals are depicted.

Explanation In order to get more insights into the plan, the user can ask questions about it. More specifically, he

can select a subset P_N of the plan properties not satisfied by the current plan. The selection is then interpreted as the question: "Why are the plan properties in P_N not satisfied?". The interface to ask a question is depicted in Figure 8. The resulting question would be: "Why does the blue truck not visit the cafe?"

The answer is then provided as shown in Figure 9. It is a list of all MUGS containing one of the plan properties in the question. Within the answer, only plan properties which are currently satisfied by the plan (enforced or by chance) are considered. The answer depicted in Figure 9 can be summarized as: "If the blue truck visits the cafe, then either package 3 can not be delivered or package 2 and 3 are not transported with the same truck."

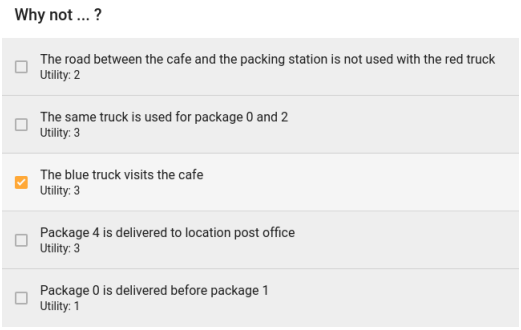

Figure 8: Interface to ask a question.

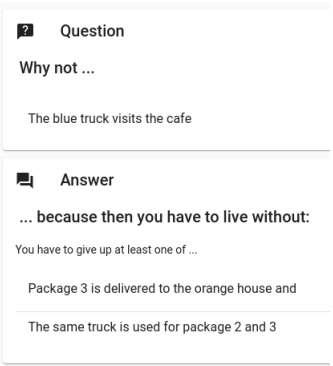

Figure 9: Answer to the question of Figure 8.

As a question depends on a specific plan it is not possible to ask a question about an unsolvable selection of hard goals.

User Study Support To allow a layperson user to understand the planning task, the plan properties and the generated plans more easily, we implemented the following adaptations. The initial state of the planning task is visualized with an image. The plan actions can be substituted with domain dependent natural language sentences. Additionally, a domain dependent animation for each plan can be added. Also the natural language representation of plan properties can be task dependent. In the example this allows to take the map of the task into account, instead of using generic location names.

To conduct a user study with a feasible amount of delay and robustness there exits the possibility to compute a *demo* from a project. To do so, the available plan properties have to be fixed and all MUGS are precomputed. To further reduce delay it could also be possible to precompute all plans. Due to the large number of combinations of hard goals, this approach might oftentimes not be feasible and is therefore not implemented by the tool.

Given such a demo, the iterative planning workflow can be further restricted for a user study: The possibility to ask questions can be enabled or disabled. Furthermore, the amount of time and/or the number of iterations available to the user can be restricted. In addition, the size of the question a user can ask can be limited.

4 User Study

We evaluated the usefulness of the explanations provided by the framework of Eif20 in the setting of iterative planning in a small preliminary user study.

User Preferences

The main purpose of the tool and the explanations is to support a user in the process of finding and understanding a plan which satisfies *his preferences*. In many domains only an expert user in the field has such intrinsic preferences. But such test persons are very difficult to recruit. In addition, the task size we can support is far to small to be interesting to a user which is used to deal with real life problems. To conduct a reasonable user study in a crowd sourcing setting, it is necessary to provide artificial preferences to the user. Otherwise, the lack of own user preferences leads to meaningless plan properties and plan differences for the user. Therefore, we decided to assign a utility as an integer value to each plan property. This is actually a contradiction to our assumption, that user preferences are difficult to elicitat, and not expressable in a utility function. But it is essential for a successful user study, as it provides the following two features. First of all, it gives the user different preferences for the different plan properties indepentent of the domain. Additionally, it is possible to motivate the user to find good plans, reflected by a high utility. This can be achieved by rewarding the user based on the best plan utility he was able to accomplish. In order to make use of this feature, the utility of each plan property in association with its natural language description, is shown to the user as depicted in Figure 8.

Setup

The test persons were 6 planning researchers from the FAI Group of Saarland University. They were all familiar with the framework, but not with the example and they used the tool for the first time. The user study was split into three parts. First, the test persons were provided a manual of the tool and had time to understand the functionality provided by it. Then, they could familiarize themselves with the planning task in Figure 2, and the plan properties listed in Section 2. The delivery of package 0 was fixed as a global hard goal.

In the second part, each plan property was assigned a utility between 1 and 3. All test persons were asked to maximize the overall utility of the plan properties satisfied by a plan. The maximal possible utility was 14. Again, this is solely done in order to provide some extrinsic motivation.

To be able to evaluate the *usefulness* of the explanations, the test persons are split in two groups of size 3 each. Group GQ+ was allowed to ask questions, group GQ- was not. The maximum number of iterations was fixed to 10 and the maximal question size to 1.

In the third part, the test persons who could ask questions had to answer a small questionnaire with respect to the helpfulness of the provided explanations.

The questionnaire was constituted of the following questions:

1. How helpful were the provided explanations in the setting of iterative planning?
2. How helpful were the provided explanations to understand the space of plans?

The questions could be answered using a discrete linear scale reaching from 1 (Didn't help at all) to 5 (Helped a lot). Additionally, there was a free text question asking for any feedback with respect to the provided explanations and the implementation of the tool itself.

Evaluation

Given the small number of test persons the resulting quantitative evaluation has to be interpreted with care. Nevertheless, the data can give some idea of the general tendency.

Numeric Evaluation The top two graphics in Figure 10 depict the overall utility achieved per iteration per test persons with and without explanations. Based on the selection of test persons a statistical analysis is not adequate. Nevertheless, For the sake of completeness, we give the 95% confidence interval for every expected value for the utility and the number of iterations. One of the test persons in GQ- quit after 5 iterations without reaching a reasonable high utility. Therefore, we will evaluate GQ- without the blue test person. The test persons in GQ-, on average, achieved a plan utility of 8.5 $[7.69, 9.30]$ (7.3, when including the blue test person). Both test persons in GQ- used all 10 iterations (when including the blue test person, on average 8.3). All test persons in GQ+ achieved a maximal plan utility of 9. They came up with their final plan within 6.3 $[4.63, 8.03]$ iterations on average. The average number of questions asked in total was 4. The bottom graphic in Figure 10 depicts the number of questions per iteration per test person. Although, the strategy of all test persons varied a lot, the questions helped to achieve a higher utility with less iterations. In addition, being able to ask questions led to a smaller amount of unsolvable tasks.

The results for both questions in the questionnaire are on average 3.66 with a variance of 0.24 on a scale from 1 (Didn't help at all) to 5 (Helped a lot).

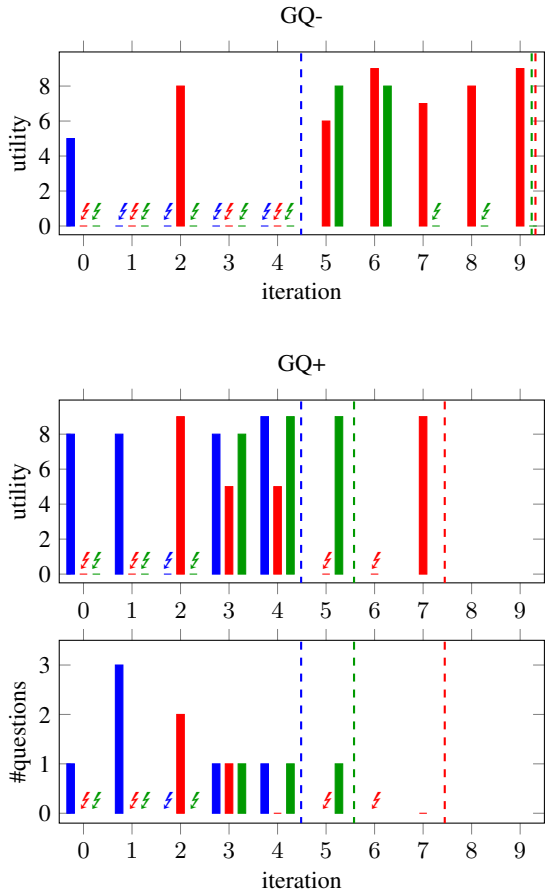

Figure 10: Plan utility and #questions per iteration. The different colors correspond to the different test persons. Dashed lines indicate max number of used iterations. ↯ indicates that the selected hard goals were unsolvable in this iteration.

Free Text Feedback In the following we evaluate and summarize the feedback given in the free text question. In general, the test persons found the possibility tho ask questions useful, as it allowed them to select plan properties more strategically.

In order to expose the functionality provided by the framework in an even more user friendly way, the interface should be further improved. For example, all participants agreed, that the interface displaying the answer needs some improvement. Especially, in situations where the answer represents multiple MUGS it has to be arranged more clearly. Some proposed to use a small guided introduction task to clearly outline the task and how the explanations could help the user.

One test person stated, that the answer should only point out the plan properties which are enforced in the plan and not only satisfied by chance.

In this user study the question size was fixed to one although larger questions are possible in general. All participants agreed, that in some cases larger questions with a size up to three would be more useful. In this case, the interface

representing the answer would have to clearly indicate the influence of the different plan properties in the question.

Almost all test persons started with an unsolvable selection of hard goals. The questions we currently support are all based on a given plan. This has the effect that the functionality of asking questions can only be exploited after a plan for a solvable selection of hard goals is found. As a consequence, most of the test persons complained, that the possibility of asking a question like "Why is this selection of hard goals unsolvable?" was missing. Although the tool does not support asking such questions, the framework of Eif20 is capable of answering them. The question can be seen as a special case of goal exclusion $\bigwedge_{g \in X} g$ entails $\neg \bigwedge_{g \in Y} g$, where X is empty. In general the answer is constituted of all MUGS $\subseteq G^{\text{hard}}$. In an iterative planning setting an explanation focusing on the latest changes leading to unsolvability, allows the user to precisely evaluate his decision. Those are the MUGS $\subseteq G_i^{\text{hard}}$ which contain at least one plan property $p \in G_i^{\text{hard}} \setminus G_{i-1}^{\text{hard}}$, where G_i^{hard} are the hard goals in iteration i.

5 Conclusion and Future Work

Eif20 introduced a framework using plan properties and their dependencies to compute plan space explanations. We implemented a Web tool using this framework in the setting of iterative planning as proposed by (Smith 2012). Additionally, we conducted a small user study showing that in general the provided questions are useful in the setting of iterative planning.

Our next goal is to run a user study with more test persons outside the community. One of the challenges will be the selection of domains, that are suitable for the target audience of test persons. Besides the logistics domain we plan to use a mars rover mission control domain as an other example from the planning benchmarks. Idealy, a test person has its own preferences with respect to the domain. Consider for example students, which have to schedule their courses for the next term as used by (Grover et al. 2020). As a domain a vast majority of people is familiar with, we intent to use the task of driving home from work and taking care of some household duties like shopping or disposing the waste paper. We will also consider further metrics proposed by (Hoffman et al. 2018) to evaluate the explanation framework.

One interesting question for the future is the combination with the framework of (Krarup et al. 2019). Their framework is based on explanations based on examples. The combination of both frameworks will allow us to evaluate the better explanation for specific situations and to combine them accordingly.

References

Baier, J. A., and McIlraith, S. A. 2006. Planning with first-order temporally extended goals using heuristic search. In *Proc. AAAI*, 788–795.

Cashmore, M.; Collins, A.; Krarup, B.; Krivic, S.; Magazzeni, D.; and Smith, D. 2019. Towards explainable AI planning as a service. In *ICAPS XAIP*.

Domshlak, C., and Mirkis, V. 2015. Deterministic oversubscription planning as heuristic search: Abstractions and reformulations. *JAIR* 52:97–169.

Edelkamp, S. 2006. On the compilation of plan constraints and preferences. In *ICAPS*, 374–377.

Eifler, R.; Cashmore, M.; Hoffmann, J.; Magazzeni, D.; and Steinmetz, M. 2020a. A new approach to plan-space explanation: Analyzing plan-property dependencies in oversubscription planning. In *AAAI*.

Eifler, R.; Steinmetz, M.; Torralba, A.; and Hoffmann, J. 2020b. Plan-space explanation via plan-property dependencies: Faster algorithms & more powerful properties. In *IJCAI*.

Fox, M.; Long, D.; and Magazzeni, D. 2017. Explainable planning. In *ICAI XAI*.

Grover, S.; Sengupta, S.; Chakraborti, T.; Mishra, A. P.; and Kambhampati, S. 2020. Radar: automated task planning for proactive decision support. *Human–Computer Interaction* 1–26.

Helmert, M. 2006. The Fast Downward planning system. *JAIR* 26:191–246.

Hoffman, R. R.; Mueller, S. T.; Klein, G.; and Litman, J. 2018. Metrics for explainable ai: Challenges and prospects. *arXiv preprint arXiv:1812.04608*.

Krarup, B.; Cashmore, M.; Magazzeni, D.; and Miller, T. 2019. Towards model-based contrastive explanations for explainable planning. In *ICAPS XAIP*.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. *The PDDL Planning Domain Definition Language*. The AIPS-98 Planning Competition Comitee.

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

Smith, D. E. 2004. Choosing objectives in over-subscription planning. In *ICAPS*, 393–401.

Smith, D. 2012. Planning as an iterative process. In *AAAI*, 2180–2185.