Towards Providing Explanations for AI Planner Decisions

Rita Borgo, Michael Cashmore, Daniele Magazzeni
King's College London
firstname.lastname@kcl.ac.uk

Abstract

In order to engender trust in AI, humans must understand what an AI system is trying to achieve, and why. To overcome this problem, the underlying AI process must produce justifications and explanations that are both transparent and comprehensible to the user. AI Planning is well placed to be able to address this challenge.

In this paper we present a methodology to provide initial explanations for the decisions made by the planner. Explanations are created by allowing the user to suggest alternative actions in plans and then compare the resulting plans with the one found by the planner. The methodology is implemented in the new XAI-PLAN framework.

1 Introduction

Artificial Intelligence technologies are increasingly ubiquitous in modern society and have the potential to fundamentally change all aspects of our lives. At the same time, increasing concerns are being raised about the transparency and accountability of AI systems.

The European Parliament resolved in 2017 to require AI systems to follow a principle of transparency, meaning they should be able to justify their decisions in a manner comprehensible to a human. Similarly, legal measures are being adopted to provide individuals affected by automated decision-making with a "right to explanation", as referred to in the recent EU General Data Protection Regulation (GDPR), in place from May 2018.

In order to engender trust in AI, humans must understand what an AI system is trying to achieve, and why. To overcome this problem, the underlying AI process must produce justifications and explanations that are both transparent and comprehensible to the user.

In 2016 DARPA launched the Explainable AI program, and since then the AI research community has been looking at this challenge with growing interest[1], mainly trying to explain or verify neural networks. While much current attention is focussed on the recent advancements in data-driven AI (e.g., Machine Learning, Deep Learning), model-based AI such as Planning is well placed to be able to address the challenges of transparency and explainability.

In AI Planning, most previous work within the realm of explainability aims to help humans understand the elements of a plan suggested by the system (e.g. [Sohrabi *et al.*, 2011],). This involves the transformation of planner output into forms that non-expert users can understand and the description of causal and temporal relations between plan steps. However, describing how a single plan works is a different problem than explaining why the planner suggests a particular plan. This requires new fundamental algorithms for generating explanations. Now, planning is being used in new critical domains (e.g., smart grid [Thiébaux *et al.*, 2013; Piacentini *et al.*, 2016], urban/air traffic control [Vallati *et al.*, 2016; Morris *et al.*, 2015], mining [Lipovetzky *et al.*, 2014], underwater robotics [Cashmore *et al.*, 2018]) and handles temporal constraints, numeric resources and continuous change [Piotrowski *et al.*, 2016], resulting in more complex plans. In this regard, explanation-generating algorithms will be instrumental in developing more robust systems for use in these critical domains.

In [Fox *et al.*, 2017] a roadmap for addressing *Explainable Planning* is proposed, which identifies a list of questions that planning systems should be able to answer, both offline before the plan is approved as well as online during plan execution. In this paper we tackle some of these questions, and in particular we focus on what we think are the most common questions a user would ask when confronted with a plan: "*Why would you do this rather than that?*", and "*Why is what you suggest more efficient than something else?*". Confronted with a question or an alternative indicated by a human user, the explanation should be a demonstration that the alternative would prevent the generation of a valid plan, or at least be no better than the existing plan. This would be a justification for the choices made by the planner. An important side effect of such an approach is that this interaction between the user and the planner enhances mixed-initiative planning, and it might be the case that the suggestion made by the user actually improves the final plan. When times and real numbers are taken into account, as it is the case in many real-world scenarios, one cannot expect optimal plans from a planner, and hence the knowledge of the domain expert should be considered in the planning process.

In this paper we present a methodology to provide expla-

[1] See Workshops on Explainable AI at IJCAI-17 and IJCAI-18.

nation for the decisions made by the planner. Explanations are created by allowing the user to explore alternative actions in the plans and then compare the resulting plans with the one found by the planner, using different metrics. We implemented the methodology in the new XAI-PLAN framework. The methodology is domain-independent and is agnostic about the planning system used. We evaluated the framework in a number of domains using ROSPlan [Cashmore *et al.*, 2015].

The paper is structured as follows. In Section 2 we provide a brief overview of related work. In Section 3 we present the methodology and in Section 4 we demonstrate the methodology with a working example. In Section 5 we describe the implementation of the XAI-PLAN framework, and Section 6 concludes the paper.

2 Related Work

Plan Explanation is an area of planning where the main goal is to help humans understand the plans produced by the planners (e.g., [Sohrabi *et al.*, 2011; Seegebarth *et al.*, 2012; Bidot *et al.*, 2010]). This involves the translation of plans to forms that humans can easily understand and the design of interfaces that help this understanding. Relevant works in robotics include [Hayes and Shah, 2017; Rosenthal *et al.*, 2016]. Similar work focuses on generating diverse solutions when user preferences are not known [Nguyen *et al.*, 2012], or top-k solutions [Katz *et al.*, 2018]. While providing the user with more choice, this does not necessarily provide explanation, nor prevent the user asking why a particular plan is selected.

Kambhampati and his team focus on the important scenario where humans and the agent have different models of the world. Explanations in that context must handle the issue of *model reconciliation* [Chakraborti *et al.*, 2017; Sreedharan *et al.*, 2018]. In the same context, *Plan Explicability* [Zhang *et al.*, 2017] focuses on human interpretation of plans. In this stream of works, the focus is on optimal plans in classical planning, which might differ because of the different models used to generate them. We focus on more expressive domains where the model is well defined, but the resulting state space is too vast and complex. In cases where the model is sufficiently complex, it is not possible to provide explanations that can be well understood in the form of model reconciliation. In this line of research, relevant works include [Smith, 2012; Langley *et al.*, 2017; Fox *et al.*, 2017].

3 Explanations for Planner Decisions

When confronted with a plan generated by a planner, a common question the user might ask is: "*Why does the plan contain this action rather than this other action that I would expect?*". Indeed, it should be noted that the support of AI is even more relevant when the AI suggests to do something *different* from what the user would do (and in particular a domain expert). At the same time such a different action plan would need an explanation before the user can be confident on its effectiveness and approve the plan.

For an effective explanation, rewriting the steps of the planning algorithm in natural language is not what is required. Nor is it very helpful to provide the heuristic evaluations of the states selected by the planner when searching for a plan [Fox *et al.*, 2017].

Rather, we argue that what can justify the selection of a set of actions by a planner is that this set of actions proves to be better, or at least no worse, than the set of actions the user would select. To this aim, a framework for providing explanations should allow the user to explore alternative actions in the plan and then compare the resulting plans with the one found by the planner. Different metrics can be used to evaluate the quality of different plans. Such an approach would increase the confidence of the user in the planner and would give him/her evidence for accepting or rejecting plans.

In this work, we use this approach to tackle the first three questions considered in the roadmap for explainable planning proposed in [Fox *et al.*, 2017]:

- Q1 Why did you do that?
- Q2 Why didn't you do something else (that I would have done)?
- Q3 Why is what you propose to do more efficient than something else (that I would have done)?

In order to evaluate different alternatives, it is necessary to infer by what metric the alternatives are to be compared (one plan might be longer but cheaper than a second — depending on the relative values of time and money, either plan might be considered better). Furthermore, the user might want to change more than one action in the current plan, or iteratively revise the plan, or explore more than one alternative for a given action. This is represented by the diagram in Figure 1 proposed in [Fox *et al.*, 2017]. Finally, as the set of possible alternatives the user might consider is potentially infinite, it is necessary to drive the user in the questions he/she can ask.

We considered all these issues in the development of XAI-PLAN, which is a methodology for providing explanations according to the view described above, and also gives the name to the framework implementing such methodology.

3.1 The XAI-PLAN Methodology

XAI-PLAN is based on the idea that the user should be allowed to explore alternative plans by suggesting different actions in the plan. This paradigm provides an answer to questions like,

Why does the plan contain action A rather than action B (that I would expect)?

The planning system is then used not only to generate the initial plan, but also to explore the alternative plans resulting from the user suggestions.

More formally, the XAI-PLAN methodology is presented in Algorithm 1. This algorithm takes as input an initial set of plans, which at the beginning contains only a single plan. The user selects an action in the plan (line 2), and this corresponds to *action A* in the question template above. As said before, the part of the question *"rather than action B"* is open to a possibly infinite set of instances. To this end, XAI-PLAN restricts this set to the applicable actions (line 3). The list of applicable actions is generated in the following way: first the state in which action a is applied in plan π is obtained. Then,

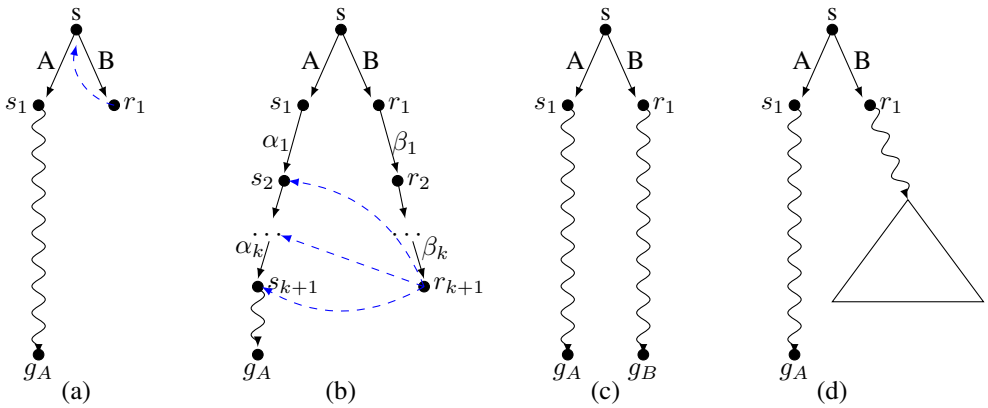

Figure 1: Possible plan behaviours after human-decision injection.

the set of all ground actions are filtered to only those whose preconditions are achieved in that state, minus the original action a. These actions are presented to the user, who can then select one of them (line 4).

Algorithm 1: XAI-PLAN

Input: initial set of plans Π
Loop
  ▷ User chooses an existing plan
1 : $\pi \leftarrow selectPlan(\Pi)$
  ▷ User chooses an action within that plan
2 : $a \leftarrow selectAction(\pi)$
  ▷ Generate list of alternative actions
3 : $applicable\_actions \leftarrow generateActions(\pi, a)$
  ▷ User selects an alternative action
4 : $a' \leftarrow selectAction(applicable\_actions)$
  ▷ New plan is generated
5 : $\pi' \leftarrow generatePlan(\pi, a, a')$
6 : $\Pi \leftarrow \Pi \cup \pi'$
EndLoop

Given a suggested action, a new plan is generated to answer the user's query (line 5). This can be done in one of four ways (described more in detail in the next section): planning from the initial state and forcing the user action to be performed, forcing the user action to be performed within a time-window, planning from the state after applying the user action, or planning both plan segments before and after the user action separately.

Finally, the new plan is added to the list of plans Π, which can be compared, and selected for further modifications (line 6). This allows iterative exploration of alternative plans.

3.2 Exploring Alternative Plans

After the user selects an alternative action, one way to explore alternative plans is to inject the user action in the plan and then replan from there. Figure 1 shows the possible outcomes:

(a) One possible behaviour is that the planner simply undoes the effect of the user action in order to return as quickly as possible to the original plan. While this might be the most efficient solution, it is undesirable from the standpoint of plan explanation, as it does not show clearly if an alternative exists, and the comparative quality of that alternative plan.

(b) The reversal of the user action can be avoided by enforcing the planner does not revisit state s. The second behaviour illustrates that the new plan, through actions $B, \beta_1, \ldots, \beta_k$ does return to the original plan, by a more or less efficient route than actions $A, \alpha_1, \ldots, \alpha_k$.

(c) The third behaviour shows the case where a new plan is found by the planner, without returning to the original planned actions.

(d) The fourth behaviour shows the case where no new plan is discovered, as the planner is unable to return to the original plan, and no alternative path to the goal exists.

Note, however, that replanning after applying the user action is not the only option, as replanning from the initial state with additional constraints is also possible. In the XAI-PLAN framework, there are four implementations of $generatePlan$ in Algorithm 1. These are:

1. planning from the state after applying the user action,
2. planning from the initial state and forcing the user action to be performed,
3. planning from the initial state and forcing the user action to be performed within a time-window,
4. or planning both plan segments before and after the user action separately.

Planning from the state obtained after applying the user action is done by disallowing the undo action, and then replanning. Planning from the initial state is achieved by updating the domain model to include a new predicate, `(applied-user-action)`, which is included as an effect of a new operator `(user-action)`. The new fact achieved by the user action is added as a goal of the problem, ensuring that the action is applied at least once in the plan.

When planning again from the initial state, there is a risk that the user action is performed differently from what was intended by the user. For example, the action might be appended to the end of a complete plan, simply to

achieve the `applied-user-action` effect. In a temporal plan, the user is able to specify a time-window in which they would like the action to be performed. This is done by adding a new predicate to the domain that is a condition of the user action operator, and can be enabled and disabled by timed-initial-lieterals (TILs). For example, the fact `(applicable-user-action)` is added as a new start condition of the operator `(user-action)`. Then, two TILs are added to the problem,

`(at LB (applicable-user-action))` and
`(at UB (not (applicable-user-action)))`,

where LB is the lower bound on the time-window, and UB is the upper bound. This ensures that the action is applied in the time-window that interests the user.

The final strategy is to plan separately from the user action to the goal (the *later plan*) and from the initial state to the user action (*initial plan*). The final plan shown to the user is obtained by concatenating the initial plan, the user action, and the later plan. Planning the later plan is performed by planning from the state after applying the user action, using the original goals. Then, a new problem is generated with the same initial state as the original problem, and goal to achieve the weakest conditions of the later plan. The weakest conditions are those facts which are conditions for actions in the later plan, not already supported by effects. An example of each approach to plan generation is described in the next section.

4 Examples of Exploring Alternative Plans

In this section we provide some examples of the four approaches to generating a new plan using the user suggested action, and some discussion on the strengths and drawbacks of each option.

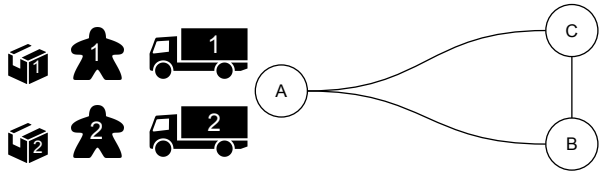

Figure 2: Problem setup for the DriverLog domain. The drivers, trucks, and packages are at location A.

Consider the problem shown in figure 2 from the *Driverlog* domain. In this problem two packages must be delivered to two separate locations. There are two drivers and two trucks available. Let us assume the plan found for this problem is shown in figure 3.

```
 0.0: (board-truck d2 t1 a) [10.0]
 0.0: (board-truck d1 t2 a) [10.0]
10.0: (load-truck p2 t1 a)  [10.0]
 10.0: (load-truck p1 t2 a) [10.0]
20.0: (drive-truck t2 a b d1) [30.0]
20.0: (drive-truck t1 a c d2) [30.0]
52.0: (unload-truck p1 t2 b)  [10.0]
52.0: (unload-truck p2 t1 c)  [10.0]
```

Figure 3: Plan generated for the driverlog problem. The highlighted action is selected by the user to be changed.

We explore the case in which the user wishes to see plans that only use one truck to deliver the package, so that they might keep one truck in reserve.

Example 1: Planning from the initial state

The user selects the highlighted action in figure 3 and chooses the alternative action, `(load-truck p1 t1 a)`, which loads the package into the first truck with the other package. A new plan must be generated from the initial state that achieves the goal and includes the action specified by the user.

This is done by updating the domain model to include a new predicate, `(applied-load-truck ?p ?t ?l)`, which is included as an effect of a new operator `(user-action-load-truck)`. The grounded fact `(applied-load-truck p1 t1 a)`, achieved by the user selected action, is added as a goal of the problem. In general, the new operator included in the domain is an exact copy of the operator selected by the user, with the addition of the `user-action` effect. The plan generated for this problem is shown in figure 4.

```
 0.0: (board-truck d2 t1 a) [10.0]
 0.0: (board-truck d1 t2 a) [10.0]
10.0: (load-truck p2 t2 a)  [10.0]
 10.0: (load-truck p1 t1 a) [10.0]
20.0: (drive-truck t2 a c d1) [30.0]
20.0: (drive-truck t1 a b d2) [30.0]
52.0: (unload-truck p1 t1 b)  [10.0]
52.0: (unload-truck p2 t2 c)  [10.0]
```

Figure 4: New plan generated from the initial state, forcing the highlighted user action.

When a time-window is specified, the user suggested `load-truck` action must be applied within a time-window, as descibed in section 3.1. The resulting plan for this problem is the same as that shown in figure 4. As we can see, the planner swaps which package to load in which truck.

The drawback of planning from the initial state is clear from this example – that the new plan still uses both trucks to deliver the packages. The question from the user, "why not use only one truck to deliver both packages?", is not given only by an action to be applied, but also the context in which that action should be applied. In this case, the action to load the package into truck `t1` in the state in which the other package was already loaded into `t1`.

Example 2: Planning after the user action

Planning from the state after applying the user action provides a plan that does show one truck delivering both packages, as shown in figure 5

```
 0.0: (board-truck d2 t1 a) [10.0]
 0.0: (board-truck d1 t2 a) [10.0]
10.0: (load-truck p2 t1 a)  [10.0]
 10.0: (load-truck p1 t1 a) [10.0]
20.0: (drive-truck t1 a b d2) [30.0]
52.0: (unload-truck p1 t1 b) [10.0]
62.0: (drive-truck t1 b c d2) [30.0]
94.0: (unload-truck p2 t1 c) [10.0]
```

Figure 5: Plan generated from the user action, using only one truck.

Consider the problem as shown in figure 6. This problem resembles the first with the addition that one driver is separated from the trucks by a long path. A plan generated from the state after the user action is shown in figure 7. The second driver, although no longer required, is still planned to walk along the long path to reach the trucks. This has the effect of dramatically reducing the plan's quality, and illustrates the drawback to planning from the user action – the quality of the plan shown to the user might be far from what is realistic. Although the plan generation is answering the question the user asks by considering the suggested action and state in which it was suggested, the answering plan is not a good answer.

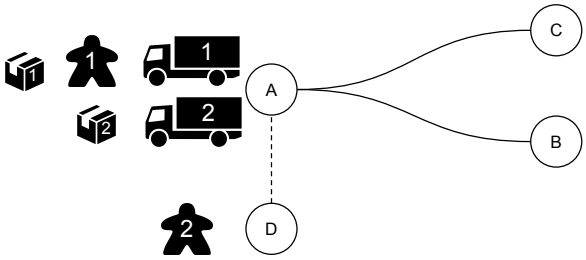

Figure 6: Problem setup for the DriverLog domain. The drivers, trucks, and packages are at location A.

```
 0.0: (walk d2 d a) [60.0]
62.0: (board-truck d2 t1 a) [10.0]
62.0: (board-truck d1 t2 a) [10.0]
72.0: (load-truck p2 t1 a)  [10.0]
72.0: (load-truck p1 t1 a) [10.0]
82.0: (drive-truck t1 a b d2) [30.0]
114.0: (unload-truck p1 t1 b) [10.0]
144.0: (drive-truck t1 b c d2) [30.0]
176.0: (unload-truck p2 t1 c) [10.0]
```
Figure 7: Plan generated from the user action, using only one truck. The second driver spends a long time walking.

Example 3: Planning before and after the user action
When planning the initial and later plans separately, the state in which the user suggested the action is not lost. The resultant plan, obtained by concatenating the initial plan, user action, and later plan is shown in figure 8. This plan uses only one truck, as the later plan was generated from the state of the user's suggested action, and also does not waste time waiting for the second driver, as the initial plan was also regenerated.

```
 0.0: (board-truck d1 t1 a) [10.0]
10.0: (load-truck p2 t1 a)  [10.0]
10.0: (load-truck p1 t1 a) [10.0]
20.0: (drive-truck t1 a b d1) [30.0]
52.0: (unload-truck p1 t1 b) [10.0]
62.0: (drive-truck t1 b c d1) [30.0]
94.0: (unload-truck p2 t1 c) [10.0]
```
Figure 8: Plan generated from the user action, using only one truck, and without waiting for the second driver.

However, this approach also has the drawback that the context of the user action might be lost. For example, if the user suggested that the first load action be altered, then the action is suggested in the context of the later plan, and not the state in which the action is suggested. Identifying exactly what context matters with respect to an action suggestion is a challenging topic that we will explore in future work.

5 The XAI-PLAN Framework

XAI-PLAN has been implemented using the ROSPlan framework [Cashmore *et al.*, 2015], which provides an interface to AI planning systems and a method for storing and modifying PDDL models in the Robot Operating System (ROS).

Figure 9 shows an overview of the XAI-PLAN framework. The XAI-Plan node implements the algorithms for generating explanatory plans, described in sections 3.1 and 4, and communicates to the user through a user interface described below. The *knowledge base*, *problem interface*, and *planner interface* are supplied by ROSPlan, which are used to store a PDDL model and provide an interface to the AI planner.

To provide a list of applicable actions to the user, the interface of the ROSPlan knowledge base is used to find the state at the point in the plan selected by the user, and then to retrieve the list of all grounded actions whose preconditions are achieved in that state.

Given an action suggested by a user, the domain is modified using the knowledge base interface, generating new predicates and operators that are required for the method of planning, as described in section 4. The initial state and goal of each problem instance are also set using this interface. The *problem interface* and *planning interface* nodes are then called to generate new plans for these problem instances.

The new plans are concatenated, if required and as described in section 4, and then shown to the user alongside a comparison with previously generated plans. The user controls these actions through a graphical user interface.

5.1 Interface Design

The interface was designed following a user-centered approach; each component either supports a user initiated action or caters for a possible user need, such as performance comparison or plan generation tracking. A screenshot of the interface is provided in Figure 10. The interface structure naturally unfolds the refinement cycles detailed in Algorithm 1.

A top bar (1) allows the user to select domain file, problem file, and planner. The *Plan* button (3), when pressed, initiates the plan generation process using the selected planner. The *Current Plan* section (2) holds the current plan, which initially will be the one generated by the planner without user intervention. Each action within the *Current Plan* window can be individually selected by the user. After selection, XAI-PLAN will prompt the planner to suggest feasible alternative actions to the user. The list of alternative actions is displayed inside the *Alternative Actions* section (4). The user is then allowed to either select any of the proposed alternative actions, or continue with the current plan and select a different action to compute a different set of alternative actions. Selection of any action within the *Alternative Actions* list will prompt the planner to generate an alternative plan which is displayed inside the *Alternative Plan* section (5).

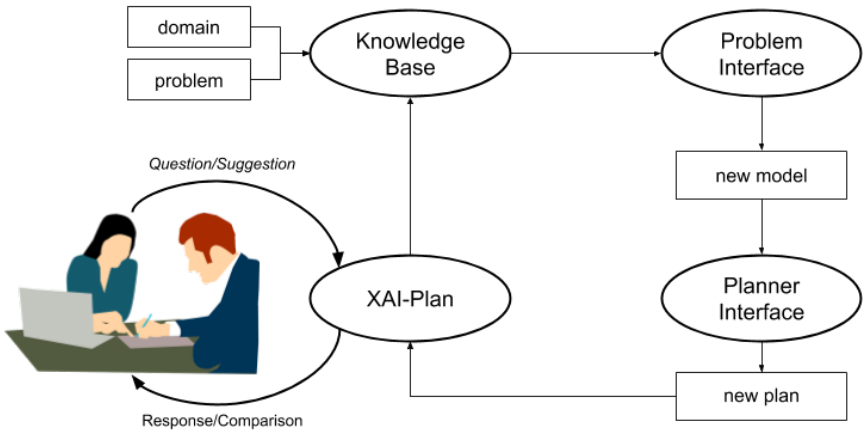

Figure 9: XAI-PLAN architecture and its integration with ROSPlan.

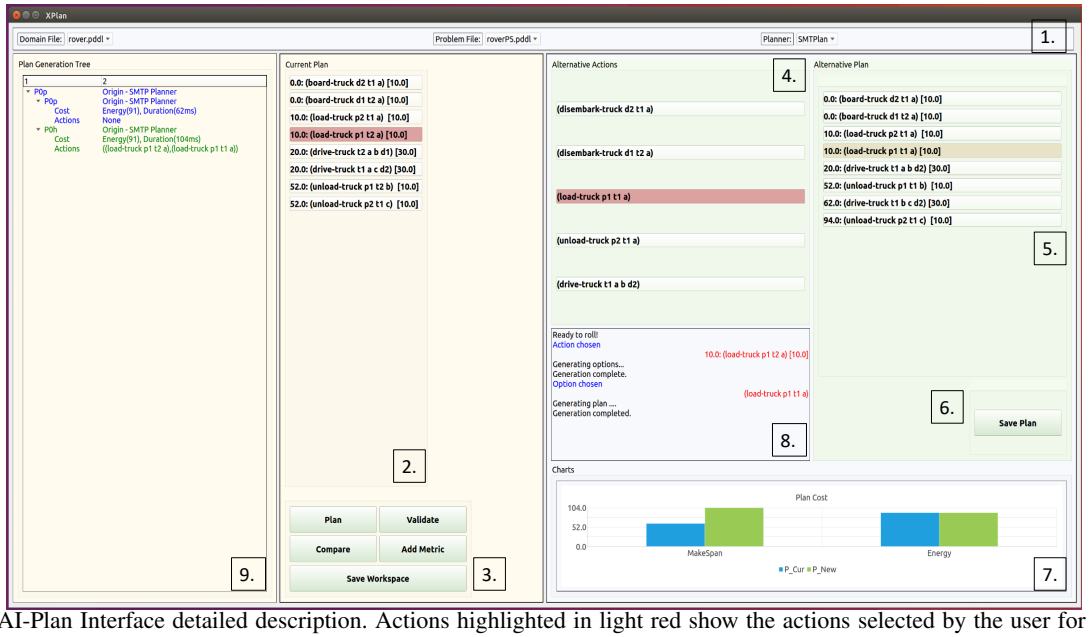

Figure 10: XAI-Plan Interface detailed description. Actions highlighted in light red show the actions selected by the user for replacement. The action highlighted in light green shows the selected alternative action within a newly generated plan.

A textbox (8) provides real-time feedback to the user, the textbox also allows user annotation to be added to the communication stream. Generated plans can be compared among each other via the *Compare* button (3). The user is allowed to define metrics to evaluate the quality of a plan via the *Add Metric* button (3). Quality of plans are displayed through a real-time chart in the *Charts* section (7). We have chosen a standard bar chart as the most appropriate representation for metric values. The *Plans Generation Tree* section (9) encodes a "plan hierarchy". The root of the hierarchy is the first plan generated by the planner. Each generated alternative plan saved by the user via the *Save Plan* button (6) is automatically inserted into the hierarchy as a child of the current plan. Each node in the hierarchy contains: the plan ID, its cost, the action suggested by the user and the action replaced. The tree allows an easy navigation between the plans the user is exploring. Finally the *Save Workspace* button (3) allows the user to save the current workspace.

6 Conclusions

We have presented a methodology for providing explanations for the decisions made by the planner. The core idea is to allow the user to explore alternative actions within plans, generate a set of new plans, and then compare their costs. An explanation for a plan found by the planner is provided by showing that the plan is no worse than the alternative plan the user would expect. The methodology is implemented in the new XAI-PLAN framework, which is integrated in ROSPlan, and provides a user-centered interface. Future work will explore a number of exciting issues such as temporal choices and context identification, and will feature a user study to assess the impact of explanations in trust and confidence.

Acknowledgements

We thank David Aha, Maria Fox, and Derek Long for their very useful comments and suggestions.

References

[Bidot *et al.*, 2010] Julien Bidot, Susanne Biundo, Tobias Heinroth, Wolfgang Minker, Florian Nothdurft, and Bernd Schattenberg. Verbal plan explanations for hybrid planning. In *MKWI*, 2010.

[Cashmore *et al.*, 2015] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, Bram Ridder, Arnau Carrera, Narcis Palomeras, Natalia Hurtos, and Marc Carreras. ROSPlan: Planning in the robot operating system. In *ICAPS*, 2015.

[Cashmore *et al.*, 2018] Michael Cashmore, Maria Fox, Derek Long, Daniele Magazzeni, and Bram Ridder. Opportunistic planning in autonomous underwater missions. *IEEE Trans. Automation Science and Engineering*, 15(2):519–530, 2018.

[Chakraborti *et al.*, 2017] T. Chakraborti, S. Sreedharan, Y. Zhang, and S. Kambhampati. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *IJCAI*, 2017.

[Fox *et al.*, 2017] Maria Fox, Derek Long, and Daniele Magazzeni. Explainable planning. *IJCAI-17 Workshop on Explainable AI*, 2017.

[Hayes and Shah, 2017] Bradley Hayes and Julie A. Shah. Improving robot controller transparency through autonomous policy explanation. In *HRI*, 2017.

[Katz *et al.*, 2018] Michael Katz, Shirin Sohrabi, Octavian Udrea, and Dominik Winterer. A novel iterative approach to top-k planning. In *ICAPS*, 2018.

[Langley *et al.*, 2017] Pat Langley, Ben Meadows, Mohan Sridharan, and Dongkyu Choi. Explainable agency for intelligent autonomous systems. In *AAAI*, 2017.

[Lipovetzky *et al.*, 2014] Nir Lipovetzky, Christina N. Burt, Adrian R. Pearce, and Peter J. Stuckey. Planning for mining operations with time and resource constraints. In *ICAPS*, 2014.

[Morris *et al.*, 2015] Robert Morris, Mai Lee Chang, Ronald Archer, Ernest Vincent Cross II, Shelby Thompson, Jerry L. Franke, Robert Christopher Garrett, Waqar Malik, Kerry McGuire, and Garrett Hemann. Self-driving aircraft towing vehicles: A preliminary report. In *AAAI Workshop on AI for Transportation*, 2015.

[Nguyen *et al.*, 2012] Tuan Nguyen, Minh Do, Alfonso Gerevini, Ivan Serina, Biplav Srivastava, and Subbarao Kambhampati. Generating diverse plans to handle unknown and partially known user preferences. *Artificial Intelligence*, 2012.

[Piacentini *et al.*, 2016] Chiara Piacentini, Daniele Magazzeni, Derek Long, Maria Fox, and Chris Dent. Solving realistic unit commitment problems using temporal planning: Challenges and solutions. In *ICAPS*, 2016.

[Piotrowski *et al.*, 2016] Wiktor Mateusz Piotrowski, Maria Fox, Derek Long, Daniele Magazzeni, and Fabio Mercorio. Heuristic planning for PDDL+ domains. In *Proceedings of IJCAI*, 2016.

[Rosenthal *et al.*, 2016] Stephanie Rosenthal, Sai P. Selvaraj, and Manuela M. Veloso. Verbalization: Narration of autonomous robot experience. In *IJCAI*, 2016.

[Seegebarth *et al.*, 2012] Bastian Seegebarth, Felix Müller, Bernd Schattenberg, and Susanne Biundo. Making hybrid plans more clear to human users - A formal approach for generating sound explanations. In *ICAPS*, 2012.

[Smith, 2012] David Smith. Planning as an iterative process. In *AAAI*, 2012.

[Sohrabi *et al.*, 2011] Shirin Sohrabi, Jorge A. Baier, and Sheila A. McIlraith. Preferred explanations: Theory and generation via planning. In *AAAI*, 2011.

[Sreedharan *et al.*, 2018] Sarath Sreedharan, Tathagata Chakraborti, and Subbarao Kambhampati. Handling model uncertainty and multiplicity in explanations via model reconciliation. In *ICAPS*, 2018.

[Thiébaux *et al.*, 2013] Sylvie Thiébaux, Carleton Coffrin, Hassan L. Hijazi, and John K. Slaney. Planning with MIP for supply restoration in power distribution systems. In *IJCAI*, 2013.

[Vallati *et al.*, 2016] Mauro Vallati, Daniele Magazzeni, Bart De Schutter, Lukás Chrpa, and Thomas Leo McCluskey. Efficient macroscopic urban traffic models for reducing congestion: A PDDL+ planning approach. In *AAAI*, 2016.

[Zhang *et al.*, 2017] Y. Zhang, S. Sreedharan, A. Kulkarni, T. Chakraborti, H. Zhuo, and S. Kambhampati. Plan explicability and predictability for robot task planning. In *ICRA*, 2017.