Towards Explainable Multi-Objective Probabilistic Planning

Roykrong Sukkerd
Institute for Software Research
Carnegie Mellon University
Pittsburgh, PA, USA
rsukkerd@cs.cmu.edu

Reid Simmons
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA
reids@cs.cmu.edu

David Garlan
Institute for Software Research
Carnegie Mellon University
Pittsburgh, PA, USA
garlan@cs.cmu.edu

ABSTRACT

Use of multi-objective probabilistic planning to synthesize behavior of CPSs can play an important role in engineering systems that must self-optimize for multiple quality objectives and operate under uncertainty. However, the reasoning behind automated planning is opaque to end-users. They may not understand why a particular behavior is generated, and therefore not be able to calibrate their confidence in the systems working properly. To address this problem, we propose a method to automatically generate verbal explanation of multi-objective probabilistic planning, that explains why a particular behavior is generated on the basis of the optimization objectives. Our explanation method involves describing objective values of a generated behavior and explaining any tradeoff made to reconcile competing objectives. We contribute: (i) an *explainable* planning representation that facilitates explanation generation, and (ii) an algorithm for generating *contrastive justification* as explanation for why a generated behavior is best with respect to the planning objectives. We demonstrate our approach on a mobile robot case study.

CCS CONCEPTS

• **Software and its engineering** → *Formal language definitions*;

KEYWORDS

Explainable Planning, Probabilistic Planning, Multi-Objective Planning

ACM Reference Format:
Roykrong Sukkerd, Reid Simmons, and David Garlan. 2018. Towards Explainable Multi-Objective Probabilistic Planning. In *SEsCPS'18: SEsCPS'18:IEEE/ACM 4th International Workshop on Software Engineering for Smart Cyber-Physical Systems , May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3196478.3196488

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SEsCPS'18, May 27-June 3, 2018, Gothenburg, Sweden
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5728-9/18/05. . . $15.00
https://doi.org/10.1145/3196478.3196488

1 INTRODUCTION

CPSs in many domains must perform tasks (i.e., take sequences of actions to achieve some goals) while optimizing for multiple quality objectives and operating under uncertainty. Use of automated planning, particularly, multi-objective probabilistic planning, to synthesize behavior of such systems can play an important role in engineering smart CPSs. However, the reasoning behind automated planning is typically opaque to the end-users. They may not understand why a particular behavior is generated, and therefore they may not be able to calibrate their confidence in the systems working properly.

One approach to solve this problem is to improve transparency and understandability of automated planning via verbal explanation. In this work, we focus on Stochastic Shortest Path Problem (SSP) as a probabilistic planning framework, and we employ multi-attribute utility theory to handle multiple optimization objectives of the planning. Our research aims to enable autonomous systems to explain their decision actions yielded from multi-objective SSP planning. This is to justify why they made certain decisions, in a formally-grounded, but human-understandable, way.

There are several challenges for human observers to understand solution policies from such a planning paradigm. The key underlying challenge from which others stem is that the choice of an optimal action at each state depends not only on the immediate consequence of that action, but also on the consequences of all possible future states and actions. From this, other difficulties follow. As a first step towards understanding a particular solution policy, human observers must know what the policy achieves in terms of the optimization objectives. However, that is non-trivial to determine manually. Moreover, since some optimization objectives may conflict, the observers also need to understand the preference structure on the optimized attributes and how conflicting objectives are reconciled under uncertainty. This is again non-trivial to determine manually, since it involves inspecting and interpreting the multi-attribute utility model underlying the cost function in SSP. Such model may not even be readily available to the observers. Furthermore, to understand what factors influence a particular optimized attribute, and how, is challenging. That such factors are stochastic makes the matter more complicated.

To address these challenges, we propose an approach to automatically justify decision actions yielded from a multi-objective probabilistic planning problem, in terms of the underlying domain semantics of the optimization objectives, and how *specifically* the competing objectives are reconciled if any. More concretely, our contributions are:

Explainable planning representation. A planning problem definition language that preserves the domain semantics of the quality optimization objectives of a planning problem, by explicitly representing the analytic models for evaluating the quality attributes, and the multi-attribute utility model on them. A planning problem specified in this language can be translated to an equivalent problem in another representation with SSP semantics. Thus, it can be solved by any SSP solver. We use an *explainable* planning problem definition, together with its solution policy, as input to our *policy justification* algorithm.

Contrastive quality-attribute-based justification. A method for generating a verbal argument of how a solution policy is preferred to other rational alternatives. Such argument contrasts the solution policy to some Pareto optimal alternative policies based on their objective values, and explains *quantitatively* the necessary value tradeoffs made to balance the competing objectives.

We demonstrate our approach on a mobile robot example.

2 PLANNING FRAMEWORK AND ASSUMPTIONS

We give an overview of the framework for multi-objective probabilistic planning, for which we propose our explainable planning approach. We also discuss the preference assumptions on the multiple quality objectives of planning, on which our current explanation approach relies.

Stochastic Shortest Path Problems. We formulate a multi-objective probabilistic planning problem as a Stochastic Shortest Path (SSP) problem [1]. We focus on a solution to an SSP $\langle \mathcal{S}, s_0, \mathcal{G}, \mathcal{A}, \mathcal{P}r, \mathcal{C} \rangle$ that is an *optimal policy* $\pi^* : \mathcal{S} \to \mathcal{A}$, which is a proper policy that minimizes the expected cumulative cost of reaching a goal state from s_0, over all closed policies. The optimal value function V^* represents the minimal of such cost to reach a goal state from each state s.

To handle multiple optimization objectives, we use a non-negative cost value $\mathcal{C}(s, a, s')$ to characterize *all* quality attributes (QAs) of concern of each 1-step decision. In this paper, we focus on SSP planning problems that aim to minimize *each* expected cumulative value characterizing *each individual* QA to reach a goal – potentially with tradeoffs. To this end, we need to make certain assumptions about the preference structure on the multiple QAs.

Preference Assumptions. We employ multi-attribute utility theory [7] as a basis for decision making to reconcile conflicting QA optimization objectives under uncertainty. We first consider the problem of minimizing all individual expected sums of the values characterizing the individual QAs at each step (s, a, s') throughout a policy. In other words, this problem assumes risk-neutral attitude towards the cumulative QA values of a policy. To ensure that SSP framework is appropriate for this type of problem, we must make the following assumptions about QA preferences.

In SSP framework, the utility of a policy is negatively proportional to the expected sum of (un-discounted) costs. The utility function of taking action a in state s, and thereafter acting optimally, is in the form:

$$u(s,a) = -\beta \cdot \Big[\min_{a'} \sum_{s'} \mathcal{P}r(s'|s,a) \big[\mathcal{C}(s,a,s') + u(s',a') \big] \Big], \quad (1)$$

where $\beta > 0$.

We characterize a policy, starting in state s and taking action a, and acting optimally thereafter, using n attributes: $X_1^{s,a}, ..., X_n^{s,a}$, where $X_i^{s,a}$ is the expected sum of the values $q_i^{s,a,s'}$ characterizing each QA i at each step (s, a, s') along the policy. Let $x_i^{s,a}$ designate a specific value of $X_i^{s,a}$. That is, we have an n-attribute utility function $u(x_1^{s,a}, ..., x_n^{s,a})$ for a policy starting in state s and taking action a, and acting optimally thereafter. To appropriately solve Bellman equation to find a policy that minimizes the values $x_1^{s,a}, ..., x_n^{s,a}$, we make the following assumptions:

(1) Attributes $X_1^{s,a}, ..., X_n^{s,a}$ have *additive independence* property. This means the utility function $u(x_1^{s,a}, ..., x_n^{s,a})$ has an additive form.
(2) Each single-attribute utility function $u_i(x_i^{s,a})$ is *monotonically decreasing*.
(3) *Risk-neutral attitude* towards the cumulative value of $q_i^{s,a,s'}$'s characterizing each quality attribute. It follows that the risk attitude towards each attribute $X_i^{s,a}$ is also neutral. This means each single-attribute utility function on $X_i^{s,a}$ has a linear form.

Given these assumptions, we can write an equation of an n-attribute utility function equivalent to Equation 1 as:

$$\begin{aligned} u(x_1^{s,a}, ..., x_n^{s,a}) = \sum_{s'} \mathcal{P}r(s'|s,a) \big[& k_1 \cdot f_1(q_1^{s,a,s'}) + ... + \\ & + k_n \cdot f_n(q_n^{s,a,s'}) + \min_{a'} u(x_1^{s',a'}, ..., x_n^{s',a'}) \big], \end{aligned} \quad (2)$$

where $f_i(q_i^{s,a,s'}) = a_i - b_i \cdot q_i^{s,a,s'}$ and $a_i, b_i < 0$ are constants of the single-attribute utility function on $X_i^{s,a}$, and k_i is a scaling constant of the n-attribute utility function $u(x_1^{s,a}, ..., x_n^{s,a})$.

From this, we can see that if the cost function in SSP can be written in the form:

$$\mathcal{C}(s,a,s') = k_1 \cdot f_1(q_1^{s,a,s'}) + ... + k_n \cdot f_n(q_n^{s,a,s'}), \quad (3)$$

then SSP planning framework is appropriate for minimizing the expected cumulative values of the individual QAs.

3 EXPLAINABILITY CHALLENGES

We discuss some of the challenges of understanding solution policies of multi-objective probabilistic planning.

3.1 Motivating Example

Figure 1 shows a mobile robot whose task is to drive from its current location to a goal in a building. The robot has to arrive at the goal as soon as possible, while trying to avoid collisions for its own safety and to avoid driving intrusively through human-occupied areas. The robot has access to the building's map (locations, and connections and distances between them), placement of obstacles (tables and chairs along a corridor), and the kinds of areas in the environment (public, private, and semi-public areas). The robot can determine its current

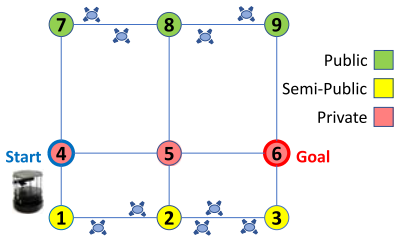


Figure 1: The robot must navigate from its current location (L1) to the goal (L9).

location, knows its current driving-speed setting, and can detect when it bumps into obstacles. The robot can also navigate between two adjacent locations, and change its driving speed between two settings. The travel time of the robot to get to a destination is determined by the distance, the robot's speed, and whether the path is partially occluded with obstacles. Driving through a partially occluded path takes longer time than an equal-length non-occluded path, and has some probability of bumping obstacles. The safety of the robot is determined by the number of collisions, which occur when the robot bumps into obstacles at full speed. Lastly, the intrusiveness of the robot is based on the kinds of areas the robot travels through. The robot is *non-intrusive*, *somewhat intrusive*, and *very intrusive* when it is in a public, semi-public, and private area, respectively.

The robot's problem can be formulated as a multi-objective SSP, where the objectives are to minimize the travel time to destination, the expected number of collisions along the way, and the cumulative penalty value for intrusiveness. Note that the travel time and intrusiveness of the robot are deterministic, unlike collisions.

3.2 Challenges

3.2.1 Value Function. Choice of an optimal action $\pi^*(s)$ at each state depends on $V^*(s)$, which depends not only on the immediate consequence of that action but also on the consequences of all possible future states and future actions. Manually inspecting them to gain insights (as discussed in the next challenges) into a particular value of $V^*(s)$ is intractable.

3.2.2 Quality-Attribute Properties. Relation from $V^*(s)$ to underlying QA properties of an optimal policy is unclear. A value $V^*(s)$ is an expected sum of values that aggregate $q_i^{s,a,s'}$'s of all QAs at each step in the policy. It is not straightforward to determine, from a value $V^*(s)$, the expected sum of $q_i^{s,a,s'}$ for each i, which characterizes the QA i of the corresponding optimal policy.

Moreover, for some QAs, an expected sum of $q^{s,a,s'}$ may not have a domain semantics associated with it. For instance, the intrusiveness levels of the mobile robot are defined by three numerical values: 0 for "not intrusive", 1 for "somewhat intrusive", and 2 for "very intrusive". An expected sum of intrusiveness values of 10.5 throughout a policy does not have a well-defined meaning.

3.2.3 Preferences and Tradeoffs. How conflicting QA optimization objectives are reconciled in an optimal policy is unclear. The preference structure over the QAs is encoded in the cost function in Equation 3. However, the functional form of a cost function cannot explain, in a quantitative way, the necessary tradeoffs among QA values to arrive at a single optimal policy, because they depend on problem instances. Such necessary tradeoffs are difficult to determine manually due to the sequential and probabilistic nature of the decision making.

Consider the possible routes to get to the goal. The robot must make tradeoff between timeliness and intrusiveness (e.g., going through route L4-L5-L6 (least time, but most intrusive) vs. route L4-L7-L8-L9-L6 (least intrusive, but more time)). It must also consider the interplay between safety and other objectives (e.g., driving at a higher speed through route segment L1-L2-L3 takes less time than driving at a lower speed; however, bumping obstacles at a higher speed has higher penalty). This matter is more complicated because bumps occur stochastically. Given the robot's generated optimal policy, it is not obvious for human observers to see what *specific, quantified* tradeoffs the robot's planning had to make to balance timeliness, intrusiveness, and safety.

3.2.4 Quality-Attribute Evaluation. What factors determine a particular QA value, and how, may be unclear. For instance, assuming that the expected travel time of a policy is known. That amount of time may be due to only the distance and speed at which the robot is traveling, or also to that the robot takes longer time because it needs to avoid obstacles along the way. To determine this manually, human observers must inspect the robot's policy, and must have access to and be able to interpret models for assessing QAs. For instance, they must figure that the robot's policy is to go through segment L1-L2-L3 at HalfSpeed, and the partially occluded corridor delays the robot by certain amount of time. This is a non-trivial task, especially if the models for assessing QAs are not readily available or not easily interpretable.

This matter is more complicated when the QA-determining factors are stochastic. Bumping into obstacles at a higher speed results in a safety penalty. However, bumping is a stochastic outcome of the robot's move action, whose probability depends on whether the pathway is partially occluded.

To address these explainability challenges, we propose: (i) an *explainable planning representation* that enables automatic explanation of the planning rationale as discussed above, and (ii) a method for generating *policy justification* based on its QA properties. However, to automatically explain how a particular QA property is determined remains our future work.

4 EXPLAINABLE PLANNING REPRESENTATION

Our approach is to use an *explainable* planning representation, which preserves the underlying semantics of QAs of a problem domain and the corresponding preference structure, to formulate a planning problem. This representation subsumes

the SSP semantics. A planning problem specified in this representation can be translated to an equivalent problem in another representation with SSP semantics. Therefore, it can be solved by any SSP solver. An *explainable* planning problem definition, together with its solution policy, can then be used as input to explanation generation. To this end, we extend the constructs of the regular SSP representation to be able to represent QA evaluation models and the corresponding multi-attribute utility model. These constructs are the following:

4.1 Compact Representation of Determining Factors of QAs

To formulate the problem using the regular SSP representation, the determining factors of QAs of a policy can be modeled as state variables (e.g., location, speed), actions (e.g., move to adjacent location), or constants associated with states and/or actions – but are not actually represented as state variables nor actions (e.g., distance between locations). Additionally, probabilistic transitions can be used to model stochastic factors of QAs (e.g., probability of the robot bumping obstacles depends on whether its path is partially occluded).

Since the determining factors of QAs can be represented in different constructs, it is not trivial to automatically parse those factors from a planning problem definition – to explain how different parts of the planning context influence the QAs, as discussed in Section 3.2.4. In this work, we use a compact representation of QA determining factors to ease the parsing.

Types for State Variables and Actions. In our proposed *explainable* planning representation, state variables and actions are typed. A state-variable type can have a set of attributes characterizing a particular state variable. Those attributes can be numerical, boolean, or finite-domain values, or other state-variable types. For instance, a state-variable type Location has two attributes: ID and Area, representing the unique ID of a particular location and the type of area that location is in (e.g., public, semi-public, private).

An action type can be parameterized by state variables, which correspond to states in which a particular action is applicable, or successor states of that action. Similar to state-variable types, action types can have attributes. However, some attributes of an action type may be *derived attributes*, which are action characteristics that depend on the state to which the action is applied. For instance, an action type MoveTo is parameterized by a Location state variable. It has two *derived* attributes: Distance and Occlusion, representing the distance and occlusion (e.g., clear, partially occluded, blocked) of the path between the source and the target locations of the action.

This is a more compact and coherent representation of the determining factors of travel time, collisions, and intrusiveness of the robot compared to the regular SSP representation. Types of state variables and actions are the building blocks for quality-attribute analytic models in our *explainable* planning representation.

Domain-Specific Vocabulary. We map types of state variables and actions to domain-specific vocabulary of concepts relating to states and actions of a particular domain. This vocabulary will be used to generate verbal explanation.

4.2 Quality-Attribute Analytic Models

As discussed in Sections 3.2.1 and 3.2.2, manually examining a value $V^*(s)$ to explain an optimal action $\pi^*(s)$ in terms of the underlying quality-attribute properties of the policy π^* is difficult. Therefore, we introduce a construct for representing QA analytic models in a planning problem definition. The construct is a function $\mathcal{Q}_i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$, referred to as *quality-attribute function* for each attribute i. A function $\mathcal{Q}_i$ characterizes the QA i at a single step (s, a, s'), based on the characteristics of the states and action (i.e., their types and attribute values). The expected sum of the values $\mathcal{Q}_i(s, a, s')$ at all steps throughout a policy then characterizes the QA i of the entire policy:

$$V_i^{\pi}(s) = \sum_{s'} \mathcal{P}r(s'|s, \pi(s)) \big[\mathcal{Q}_i(s, \pi(s), s') + V_i^{\pi}(s') \big] \quad (4)$$

There are different kinds of QAs based on how they are quantified. Thus, they shall be explained differently. In this paper, we discuss three kinds of QAs.

Counts of Events. QAs of this kind are quantified as expected numbers of occurrences of events of interest, where the objectives are to minimize occurrences of "bad" events. The robot bumping into obstacles at a higher speed is an example of such event, where the safety objective of planning is to minimize the expected number of such collisions.

An event e representing a transition (s, a, s') is a predicate over the characteristics of the states and action involved. For instance, a collision event $e_{collide}$ is a predicate $s.rSpeed > \theta_{safeSpeed} \wedge s'.rBumped = True$. Currently, we exclude events that have durations (i.e., span across multiple transitions) from the definition. A QA function $\mathcal{Q}_e$ characterizing event e is an indicator function of the set of transitions satisfying e. Thus, a value $V_e^{\pi}(s)$ in Equation 4 is an expected number of occurrences of event e of policy π starting at state s.

Standard Measurements. QAs of this kind can be measured in a standard, scientific way, either directly or by deriving from other measurements. Such QAs are quantified by their magnitudes (e.g., duration, length, rate of change, utilization, etc.), typically measured in real values. A QA function $\mathcal{Q}_i$ of this kind is a measurement model of the magnitude of i at a single step (s, a, s'), which uses the characteristics of the states and action as parameters. Recall that these magnitudes must be additive to be appropriate for the SSP planning framework. The travel time of the robot is an example of this kind, where the travel time of a move transition $\mathcal{Q}_{time}$ depends on the robot's speed and the traveling distance, with an additional delay if the path is partially occluded by obstacles. The expected travel time of the robot following policy π, starting from state s, is $V_{time}^{\pi}(s)$ in Equation 4.

Non-Standard Measurements. Some properties do not have associated standard measurements, or their exact standard measurements may not be available to the modelers.

Nonetheless, given the domain knowledge, a carefully-defined arbitrary measurement can be used to characterize policies with respect to a particular property of concern. One such *non-standard measurement* approach is to map characteristics of states and actions to numerical values representing *levels or degrees* of the property at each step (s, a, s'), and then the expected sum of these values throughout a policy characterizes the property of that policy. The intrusiveness of the robot is an example of a non-standard QA, where the intrusiveness of the robot is given the values $0, 1$, and 3 for then it moves into the public, semi-public, and private areas, respectively.

An expected sum of values representing an abstract property, such as intrusiveness, does not have a well-defined meaning. Unlike for standard measurements, it is not intelligible to communicate a value such as $V^{\pi}_{intrusive}(s)$ in an explanation. Instead, we propose to communicate the distribution of different measurement values throughout a policy in an explanation. To this end, we calculate the expected count of events characterizing each measurement value. Moreover, instead of communicating numerical measurement values, we use qualitative terms (e.g., *"not intrusive"*, *"somewhat intrusive"*, and *"very intrusive"*) to describe them in an explanation.

Vocabulary for Quality Attributes. We map QA analytic models to domain-specific vocabulary to be used to generate verbal explanation. The vocabulary includes QA names, measurement units for standard QAs, and qualitative terms for describing non-standard QAs.

4.3 Multi-Attribute Utility Model

As discussed in Section 3.2.3, manually determining the specific and quantified value tradeoffs taken by the planning to reconcile conflicting QA objectives is challenging. To enable our explanation algorithm to address this challenge, we introduce a construct for representing multi-attribute utility model on the QAs in a planning problem definition.

Recall from Section 2 that the functional form of a cost function $\mathcal{C}(s, a, s')$ is determined from all single-attribute utility functions on $X_i^{s,a}$ and an n-attribute utility function on all $X_i^{s,a}$'s (Equation 2). Our *explainable* planning representation has a construct for specifying these utility functions explicitly. This allows us to explain the preference structure on the multiple QAs quantitatively.

5 POLICY JUSTIFICATION

A policy justification argues why a particular solution policy is best with respect to the quality optimization objectives. This section describes our policy justification approach, which explains the QA properties of a solution policy and any necessary tradeoffs made to reconcile competing objectives.

5.1 Describing Objectives and Quality-Attribute Properties

The first part of a policy justification describes what QA objectives are considered, and what QA properties a solution policy has. Essentially, we explain the value $V^*(s)$ that determines each optimal action $\pi^*(s)$ in terms of the underlying QA values of π^*. This step utilizes the QA analytic models and the corresponding domain-specific vocabulary represented in an *explainable* planning problem definition.

Numerical Query. To determine the counts of events, or the QA values with standard measurements, of a solution policy, we use the corresponding $\mathcal{Q}_i$ functions and calculate the expected sums of $\mathcal{Q}_i(s, a, s')$ values throughout the policy, as shown in Equation 4. Similarly, to determine the QA properties with non-standard measurements, we query the distributions of the *non-standard measurement values* occurring in the policy, as discussed in Section 4.2.

Quality-Attribute Language Templates. To generate verbal explanation of the objectives and the QA properties of a solution policy π^*, we use predefined natural-language templates. Recall that different kinds of QAs should be explained differently. Table 1 shows examples of verbal explanation of QA objectives and properties. The italicized terms are originally placeholders in the templates. These placeholders are then filled with either terms from the domain-specific vocabulary provided in a planning problem definition, or values $V_i^{\pi^*}(s)$ for each QA i.

Kind of QA	Optimization Objective	QA Property
Count of events	"minimize the expected number of *collisions*"	"the expected number of *collisions* is *0.8*"
Standard measurement	"minimize the expected *travel time*"	"the expected *travel time* is *10 minutes*"
Non-standard measurement	"minimize the expected *intrusiveness*"	"the policy is expected to be *non intrusive* for *5* steps and *somewhat intrusive* for *2* steps"

Table 1: Examples of verbal explanation of quality-attribute objectives and properties.

5.2 Contrastive Quality-Attributed-Based Justification

The second part of a policy justification argues why a particular solution policy is generated. There are different aspects and kinds of explanation. But in this work, we focus on arguing why a solution policy is superior to other feasible alternatives, or at least not inferior to other alternatives – as different policies may have the same overall utility value. This kind of explanation is appropriate for optimization problems.

Since optimization problems in question are multi-objective, arguing why a particular solution is superior must address how competing objectives are reconciled, i.e., how tradeoffs are made, if any. To this end, we explain the tradeoffs by contrasting the solution policy to some Pareto optimal alternatives based on their QA properties. We then justify the tradeoffs according to the utility models underlying the cost function of the planning problem. We call this *contrastive quality-attribute-based justification.*

If there are no competing QA objectives, a justification indicates absolute optimality of a solution policy.

Determining Alternative Policies. Algorithm 1 outlines an approach for sampling Pareto-optimal alternative policies *nearby* the solution policy. The key idea of the approach is the

following. Starting from the QA values of the solution policy π: $x_1^{s,\pi(s)}, \ldots, x_n^{s,\pi(s)}$. For each QA i, we determine a new value x_i' that is more preferable than $x_i^{s,\pi(s)}$. Then, we construct a new planning problem with one less optimization objective (excluding that of the QA i – resulting in a new cost function of the SSP problem), and with the constraint that the QA i must be at least as good as x_i'. Next, we find an optimal, constraint-satisfying solution policy π' for the new planning problem. This policy is an alternative to the selected solution policy π for the original planning problem. We perform this step iteratively until we obtain up to M_i number of alternative policies for each i. Using this method, we can obtain up to approximately $n \cdot M_{avg}$ alternative policies; each one improves at least one QA but compromises at least one other QA.

There are ways to fine-tune this algorithm, which might lead to a more effective explanation of the tradeoff rationale of planning. Algorithm 1 searches for each alternative by improving one QA at a time. However, we may also improve multiple QAs at once, but appropriate heuristics to guide such selection and search may be needed. Furthermore, the parameters of Algorithm 1 (e.g., $\Delta u_i, \theta_i, M_i$) may impact the relevance of the generated alternatives (e.g., alternatives that have too-undesirable values in some QAs, even though Pareto optimal, may be considered unnecessary to an explanation). Investigating these issues remains our future work.

Algorithm 1: Find Pareto-optimal alternative policies

input : π: solution policy
s: initial state
$u_1, \ldots, u_n$: 1-attribute utility functions on $x_i^{s,\pi(s)}$
u: n-attribute utility function on all $x_i^{s,\pi(s)}$'s
$u_{\setminus 1}, \ldots, u_{\setminus n}$: all $(n-1)$-attribute utility functions
$\Delta u_1, \ldots, \Delta u_n$: increment sizes of utilities
$\theta_1, \ldots, \theta_n$: maximum utilities
$M_1, \ldots, M_n$: max. # of alternatives / attribute
output : A set of alternative policies Π'

$\Pi' \leftarrow \emptyset$;
$D \leftarrow$ attributes to be explored, e.g., $\{1, \ldots, n\}$;
while $D \neq \emptyset$ **do**
  $i \leftarrow$ remove an attribute from D;
  $count_i \leftarrow 0$;
  $v_i \leftarrow u_i(x_i^{s,\pi(s)})$;
  $u_{\setminus i} \leftarrow (n-1)$-attribute utility function on all $x_j^{s,\pi(s)}$'s, $j \neq i$;
  while $v_i < (\theta_i - \Delta u_i) \wedge count_i < M_i$ **do**
    // Improve utility of attribute i
    $v_i \leftarrow v_i + \Delta u_i$;
    $\bar{x}_i^{s,\pi(s)} \leftarrow$ all $x_j^{s,\pi(s)}$, where $j \neq i$;
    $\pi' \leftarrow \operatorname{argmax}_{\pi} u_{\setminus i}(\bar{x}_i^{s,\pi(s)})$, subject to $u_i(x_i^{s,\pi(s)}) \geq v_i$;
    if π' *exists* **then**
      $\Pi' \leftarrow \Pi' \cup \{\pi'\}$;
      $count_i \leftarrow count_i + 1$;
      for $j \neq i$ **do**
        if $u_j(x_j^{s,\pi'(s)}) \geq u_j(x_j^{s,\pi(s)}) + \Delta u_j$ **then**
          // Disregard attributes that were sufficiently improved in this iteration in future exploration
          $D \leftarrow D - \{j\}$

Explaining the Value Tradeoffs. Comparing the solution policy to each Pareto-optimal alternative policy demonstrates *quantitatively* the necessary value tradeoffs in planning. Our policy justification indicates the amount of gain-loss in the QAs if one were to choose each alternative over the original solution policy. It then indicates preference towards the solution policy by arguing that such gain is not worth the loss, reflecting the QA utility models underlying the cost function of planning. We use a predefined natural-language template for generating verbal justification:

"I could [(1) vp. improve these QAs to these values]*, by* [(2) vp. carrying out this alternative policy] *instead. However, this would* [(3) vp. worsen these other QAs to these values]*. I decided not to do that because* [(4) np. the improvement in these QAs] *is not worth* [(5) np. the deterioration in these other QAs].

The statement fragments (1) and (3) contrast the QA values of the original solution policy with those of an alternative policy, by describing the gains and the losses quantitatively. (2) describes the alternative policy. (4) and (5) restate the gains and the losses qualitatively, as part of rejecting the alternative policy.

6 DEMONSTRATION

We demonstrate a policy justification that our approach generates for a solution policy[1,2] from the mobile robot example.

"I aim to minimize the expected travel time, the expected number of collisions, and the expected intrusiveness. I plan to move from L4 *to* L1 *at* FullSpeed*, from* L1 *to* L3 *at* HalfSpeed*, and from* L3 *to* L6 *at* FullSpeed*. The travel time is 10 minutes, the expected number of collisions is 0, and the policy is somewhat intrusive for 3 steps and very intrusive for 1 step.*

I could decrease the travel time to 5 minutes, by moving through L4-L5-L6 *at* FullSpeed *instead. However, this would increase the intrusiveness to be very intrusive for 2 steps. I decided not to do that because the decrease in travel time is not worth the increase in intrusiveness.*

I could also decrease the intrusiveness to being non-intrusive for 3 steps and very intrusive for 1 step, by moving from L4 *to* L7 *at* FullSpeed*, from* L7 *to* L9 *at* HalfSpeed*, and from* L9 *to* L6 *at* FullSpeed *instead. However, this would increase the travel time to 15 minutes. I decided not to do that because the decrease in intrusiveness is not worth the increase in travel time."* [3]

This is because the algorithm: (i) searched to reduce time on the Pareto curve and found a policy that goes via the shortest, obstacle-free route at full speed, but through a private area, and (ii) searched to reduce intrusiveness and found a policy that goes through a public area, but has the longest route with obstacles. Since the expected collision is 0

[1] To explain why a solution policy is selected, we have to describe what the policy is first. However, we do not discuss an approach for policy description in this paper.

[2] Here we omit the term "expected" from the verbal explanation when a QA of the policy has deterministic value.

[3] Note that there are other Pareto optimal alternative policies that the justification algorithm did not pick that might clarify the tradeoff rationale further. This is a challenge of tuning the parameters of Algorithm 1, as discussed in Section 5.2.

in the original solution policy, the algorithm did not attempt to find a policy to reduce it.

7 RELATED WORK

There are several existing works that aim to explain why intelligent agents or systems produce particular behavior. Some existing works have investigated policy explanation for MDP-based planning systems. Elizalde et al. [4, 5] contributed an approach that identifies factors that are most influential to the decisions. Khan et al. [8] also contributed an approach for explaining an optimal action in a policy by computing the frequency of reaching a goal by taking the action. Hayes and Shah [6] contributed algorithms for generating descriptions of a robot's control policies and to respond to queries from humans, including questions about environmental conditions under which certain behavior will occur and vice versa, and why certain behavior did not occur.

While these works enable policy explanation for MDP-based planning, they focus on explaining optimal actions in a policy solely based on either: (i) impact of action choices on the total reward or goal reachability, or (ii) conditions of the state variables under which certain actions are taken. The works by [4, 5, 8] fall into the first category. They do not address the underlying semantics of rewards, which we argue that it is important for promoting human understanding of the decision rationale. The work by [6] falls into the second category. Their generated explanations contrast conditions under which different actions are taken, but not justify why the actions should be taken. Moreover, unlike our work, they do not address the problem of multiple optimization functions and justifying optimal actions on the basis of these objectives.

Plan explanation for other planning paradigms has also been studied. Seegebarth et al. [14] proposed an approach for explaining hybrid planning, by proving the necessity of plan steps and orderings on plan steps. Bidot et al. [2] presented an approach for generating verbal explanations of hybrid planning, which justify the ordering of two tasks and for the temporal position of a single task. Unlike these prior works, our approach does not aim to explain the causality of plan or policy steps. Instead, we focus on explaining the relative desirability of a policy, with respect to the quality optimization objectives.

Apart from plan and policy explanation, there are several other works on explaining behavior of rule-based or machine-learning-based systems. Lim and Dey et al. [3, 11, 13] contributed automatic explanation approach for the different explanation types and decision model types, including rules, decision trees, naive Bayes, and hidden Markov models [12]. Kulesza et al. [9, 10] investigated how explanations can affect end users' mental models of intelligent agents' personalized behavior, focusing on explaining decision-tree bagging ensemble classifier.

8 CONCLUSION AND FUTURE WORK

We contribute an approach for automatic explanation of the rationale behind decision actions (policies), yielded from multi-objective probabilistic planning. Such explanation argues for why a particular solution policy is superior to other feasible alternatives with respect to the optimization objectives. This includes explaining the objective values of the solution policy, and how competing objectives are reconciled, grounded on the preference structure on the QAs.

Currently, our explanation approach relies on strong assumptions about the preference structure on quality attributes. We plan to investigate approaches that are applicable to a broader class of multi-attribute utility models. We also have yet to solve the challenge of explaining the determining factors of QAs. Furthermore, we aim to address the problems of finding alternative solution policies that are deemed relevant to a contrastive justification for a particular policy – constituting a more effective explanation.

ACKNOWLEDGEMENTS. This work is supported in part by award N00014172889 from the Office of Naval Research, the Department of Defense, and is based upon research supported by (while Dr. Simmons was serving at) the National Science Foundation.

REFERENCES

[1] Dimitri P Bertsekas and John N Tsitsiklis. 1991. An analysis of stochastic shortest path problems. *Mathematics of Operations Research* 16, 3 (1991), 580–595.
[2] Julien Bidot, Susanne Biundo, Tobias Heinroth, Wolfgang Minker, Florian Nothdurft, and Bernd Schattenberg. 2010. Verbal Plan Explanations for Hybrid Planning. In *MKWI*. 2309–2320.
[3] Anind K Dey. 2009. Explanations in Context-Aware Systems. In *ExaCt*. 84–93.
[4] Francisco Elizalde, L Enrique Sucar, Manuel Luque, J Diez, and Alberto Reyes. 2008. Policy explanation in factored Markov decision processes. In *Proc European Workshop on Probabilistic Graphical Models (PGM)*. 97–104.
[5] Francisco Elizalde, Luis Enrique Sucar, Alberto Reyes, and Pablo deBuen. 2007. An MDP Approach for Explanation Generation. In *ExaCt*. 28–33.
[6] Bradley Hayes and Julie A Shah. 2017. Improving Robot Controller Transparency Through Autonomous Policy Explanation. In *Proc Int'l Conf. on Human-Robot Interaction (HRI)*. ACM.
[7] Ralph L Keeney and Howard Raiffa. 1993. *Decisions with multiple objectives: preferences and value trade-offs*. Cambridge university press.
[8] Omar Zia Khan, Pascal Poupart, and James P Black. 2009. Minimal Sufficient Explanations for Factored Markov Decision Processes. In *Proc Int'l Conf. on Automated Planning and Scheduling (ICAPS)*.
[9] Todd Kulesza, Simone Stumpf, Margaret Burnett, and Irwin Kwan. 2012. Tell me more?: the effects of mental model soundness on personalizing an intelligent agent. In *Proc SIGCHI Conf. on Human Factors in Computing Systems (CHI)*. ACM, 1–10.
[10] Todd Kulesza, Simone Stumpf, Margaret Burnett, Sherry Yang, Irwin Kwan, and Weng-Keen Wong. 2013. Too much, too little, or just right? Ways explanations impact end users' mental models. In *Proc. Int'l Symposium Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 3–10.
[11] Brian Y Lim and Anind K Dey. 2009. Assessing demand for intelligibility in context-aware applications. In *Proc Int'l Conf. on Ubiquitous Computing (UbiComp)*. ACM, 195–204.
[12] Brian Y Lim and Anind K Dey. 2010. Toolkit to support intelligibility in context-aware applications. In *Proc Int'l Conf. on Ubiquitous Computing (UbiComp)*. ACM, 13–22.
[13] Brian Y Lim, Anind K Dey, and Daniel Avrahami. 2009. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *Proc SIGCHI Conf. on Human Factors in Computing Systems (CHI)*. ACM, 2119–2128.
[14] Bastian Seegebarth, Felix Müller, Bernd Schattenberg, and Susanne Biundo. 2012. Making hybrid plans more clear to human users-a formal approach for generating sound explanations. In *Proc Int'l Conf. on Automated Planning and Scheduling (ICAPS)*.