Preference Adaptation: user satisfaction is all you need!

Nianyu Li
ZGC Laboratory
li_nianyu@pku.edu.cn

Mingyue Zhang
South West University
mingyuezhang@pku.edu.cn

Jialong Li
Waseda University
lijialong@fuji.waseda.jp

Eunsuk Kang
Carnegie Mellon University
eunsukk@andrew.cmu.edu

Kenji Tei
Waseda University
ktei@aoni.waseda.jp

***Abstract*—Decision making in self-adaptive systems often involves trade-offs between multiple quality attributes, with user preferences that indicate the relative importance and priorities among the attributes. However, eliciting such preferences accurately from users is a difficult task, as they may find it challenging to specify their preference in a precise, mathematical form. Instead, they may have an easier time expressing their displeasure when the system does not exhibit behaviors that satisfy their internal preferences. Furthermore, the user's preference may change over time depending on the environmental context; thus, the system may be required to continuously adapt its behavior to satisfy this change in preference. However, existing self-adaptive frameworks do not explicitly consider dynamic human preference as one of the sources of uncertainty. In this paper, we propose a new adaptation framework that is specifically designed to support self-adaptation to user preference. Our framework takes a human-on-the-loop approach where the user is given an ability to intervene and indicate dissatisfaction and corrections with the current behavior of the system; in such a scenario, the system automatically updates the existing preference values so that the new, resulting behavior of the system is consistent with the user's notion of satisfactory behavior. To perform this adaptation, we propose a novel similarity analysis to produce changes in the preference that are optimal with respect to the system utility. We illustrate our approach in a case study involving a delivery robot system. Our preliminary results indicate that our approach can effectively adapt its behavior to changing human preference.**

I. Introduction

Self-adaptive systems often make automated decisions that take multiple, possibly competing *quality attributes*, also known as non-functional objectives, from stakeholders into consideration. For instance, a hospital e-service system may be associated with the following ones: tangibles, reliability, responsiveness, confidence, empathy, quality of information, and integration of communication issues [1]. In certain self-adaptation frameworks, the developer may encode these *quality attributes* by defining functions, each describing the relationship between the system state and a utility value that indicates the level of satisfaction of a particular attribute [2], [3]. Building on these functions, the overall utility of the system is then defined by an aggregation function that corresponds to the weighted sum of utility values for the quality attributes [4]. During adaptation, if multiple candidate decisions are available, as is often the case in practice [5], [6], the one with the highest aggregated utility value is selected by the automated planner.

An important factor in the utility-based approach to adaptation is the *preference* that indicates the relative importance and priorities among quality attributes. One common representation of preference is a vector of weights that are assigned to the quality attributes in the aggregation function. Typically, the preference is elicited from stakeholders or users and configured into the system before the deployment [7], [3], [6], [8]. In certain applications, however, one's preference may not be static; depending on the evolving environmental context, the user's preference may also change over time. For example, for a delivery robot, efficiency (i.e., the delivery speed) may initially be considered to be the most desirable quality attribute to optimize, but as the robot enters a hazardous environment (e.g., a traffic crossing), the user may prioritize safety over efficiency. Similarly, depending on the monetary value and fragility of the object that is being carried by the robot, the importance weights assigned to safety and efficiency may also vary. In such applications, the underlying self-adaptive framework must be capable of re-adjusting its strategies in response to the evolving user preference [9], [10].

There is relatively little prior work on self-adaptive frameworks that explicitly take into account dynamically changing human preference. Existing approach (as far as we are of) supporting this type of adaptation is by allowing the user to directly manipulate the weights for different quality attributes [11], [9]. A major limitation to this approach, however, is that users typically find it challenging to express their objectives and preferences in a precise, mathematical representation (such as numerical weights) [2], [12], [13]. Even if the user considers one quality attribute to be more important than others, they may have a difficult time determining the magnitude of difference in their importance. In general, their provided weight vector may be a rough, inaccurate approximation of the user's internal preference; this could, in turn, result in the system sometimes behaving in ways that are not satisfactory to the user.

However, while they struggle with precisely specifying their

preference, users may be better at recognizing and distinguishing satisfactory system behaviors from those are not [14], [2]. Furthermore, when they recognize an unsatisfactory behavior, they may be able to correct it by providing an alternative, more desirable behavior; e.g., giving feedback to a smart light control system by turning off the light after the system turns it on against the user's preference.

In this paper, we propose a novel approach to self-adaptation that explicitly considers dynamic human preference as one of the sources of uncertainty. Our approach allows the user to dynamically update their preference without being aware of the underlying quality attribute functions or weights. The key idea behind this approach is the preference update through *human guided correction*: When the user recognizes a system behavior that is not satisfactory (in terms of how well it is achieving desired quality attributes), they provide feedback to the system by rejecting the unsatisfactory behavior or providing an alternative, more desirable behavior. Our framework then leverages a novel *similarity analysis* to infer minimal changes to the preference values that are sufficient to account for the difference between the existing and user-indicated behaviors. To realize this approach, we propose a two-layer self-adaptation framework: in the high-level layer, the system monitors for and dynamically updates the preference in response to an correction by the user. Subsequently, in the lower layer, the system will re-adjust the adaptive strategy to be executed by the automated planner based on the new, updated preference.

We have built a prototype implementation of our self-adaptive framework. As a case study, we demonstrate our approach over a delivery robot system with multiple quality attributes such as safety, efficiency, and privacy. Our preliminary experiments show that our approach can effectively adapt to changing human preference and while maintaining an optimal level of the overall system utility.

The rest of the paper is structured as follows. Section II provides a running example involving a delivery robot and describes the related work. Section III gives an overview of the proposed framework while Section IV describes the preference update algorithm based on similarity analysis. Section V presents the results from our preliminary experiments. Section VI concludes with a discussion of limitations and directions for future work.

II. Motivating Scenario and Related Work

This section introduces a motivating scenario to illustrate our approach, and related work on human-involved adaptive systems and preference oriented self-adaptation.

A. Motivating Scenario

To motivate our approach, we use the example of robot mission scenario in smart home environment where a robot performs a series of goods-moving tasks on request of users from a starting point to a goal location, as shown in Figure 1. The robot has to arrive at the destination as soon as possible, while trying to avoid collisions for its own safety and to avoid driving intrusively through user-occupied areas.

In this scenario, there are three quality attributes considered by users: *safety*, to avoid collisions with obstacles; *privacy*, to not intrude into user personal spaces; *efficiency*, the travel time of the robot when executing a good mission. The robot has an automated controller to generate an optimal path by taking the preference of quality attributes into account and then adaptively move following the path. Efficiency is initially considered to be the most desirable quality attribute to optimize, however, with the urgent work to handle by himself, user may prioritize privacy over efficiency without disturbance from the robot. Instead of precisely specifying his new preference with values, user is easier at recognizing unsatisfactory behavior and complaining when, e.g., robot enters a certain private area, and correcting the robot's movement by giving feedback to robot to avert this area. Based on user feedback, hopefully, the robot could automatically reason about user's new preference and adapt its behavior accordingly to satisfy this change.

In the setting of this example, the safety cost is 1 for each traversed occluded path segment (i.e., two triangles in Figure 1), 0.5 for each traversed semi-occluded (i.e., one triangle), and 0.1 for each without obstacles. The privacy cost is 0.5 for each semi-privacy location and 1 for each privacy location that is visited. The efficiency cost is set to the traveled distance indicated by the number of segments visited. Based on the cost c_i over a quality attribute QA_i, the utility of a path s in terms of that quality attribute is defined as $u_i(s) = 10 - c_i(s)$.

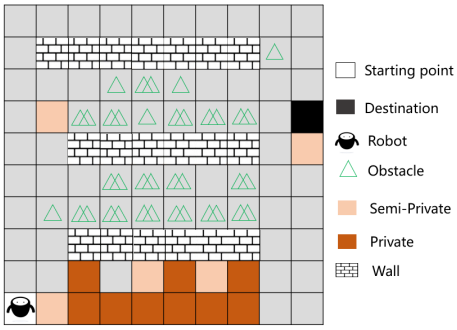


Fig. 1: Building Map for Motivating Scenario.

The overall system utility of U over a strategy (i.e., a path from the starting point to the destination) is calculated depending on the costs in terms of safety, privacy, and travel time. The path selection relies on the following aggregation function whose value should be maximized by the generated plan with preference, i.e., quality attribute weights w_i, for decision making process: $U(s) = \sum_{i=1}^{n} w_i \times u_i$. One preference condition could be $\langle$ 0.3, 0.3, 0.4 $\rangle$, i.e., the weights set to 0.3 for both safety and efficiency and 0.4 for privacy.

B. Related Work

Our framework addresses the need of self-adaptive framework to manage run-time goal adaptation specifically with user preference for interactive planning. Relevant related work

is concerned with human-involved adaptive systems and run-time adaptation of goals especially on preference and utility functions.

1) Human-Involved Adaptive systems: Automation is one desirable characteristic of self-adaptation, certain adaptive systems achieve better satisfaction of system goals when their adaptation is able to exploit human input [15]. In some cases, human can provide expertise or knowledge not available to the system and can detect problems that the system is unaware of [16], [17]. One way to achieve this synergy for the system is by placing an operator *in-the-loop* between the self-adaptation framework and the environment as a deciding authority. A variant of *human-in-the-loop* is *human-on-the-loop*, in which the operator plays a less interactive role; in this approach, he only periodically monitors the interaction between the system and the environment, and intervenes only when deemed necessary or unsatisfactory [18]. In this paper, we focus on self-adaptive systems that employ a human-on-the-loop approach where user preference is the ground truth that the system is trying to approximate and his periodical corrections on system behaviors will invoke the preference adaptation.

2) Goal Adaptation: In the run-time category, based on the literature review, few research efforts explore a goal-based model in the decision-making process [19]. For example, Kramer et al. discuss an architecture-based adaptation approach, which utilizes a goal management layer [20]. This layer generates a reactive plan to satisfy the goals. Salehie et al. have also worked on an adaptation approach based on quality goals, in order to trace and satisfy these goals at run-time [21]. In another work, Salehie et al. propose a Goal-Action-Attribute model by explicitly tracing adaptation goal at run-time which is helpful in adaptation decision making [22]. However, their work deals mostly with the deciding process by choosing among alternative actions at run-time, i.e., action selection problem. Besides, an Analytic Hierarchy Process (AHP) by pairwise comparison of goals is suggested to be used for prioritizing goals [23].

Utility functions together with user preference have been widely used for self-adaptive systems as a common mechanism in defining multiple objectives [24]. Human preference has been investigated and captured in satisfying temporal logic properties with timing constraints by weighted Signals Temporal Logic (WSTL) via weights, upon which a system behavior with incompatible (infeasible) tasks or with performance preferences can be reasoned about [25]. One preference approach is in a way similar to linguistic ranking methods by specifying one of the following characterizations for every two objectives: less important; much less important; equally important; more important; much more important; don't care. Then, these linguistic categories are transformed into real numbers that can be utilized in optimization as objective weightings with the use of (fuzzy) preference relations and induced orders [26]. Wohlrab et al. propose a lightweight method for utility function definition based on goals priorities elicitation and negotiation, leading to a consensus between multiple stakeholders [27]. Their approach, however, is intended for development rather than runtime use.

Lin et al. propose a three-level model to learn multiple inhabitant's preference in a smart home environment based on a sequence of inhabitant-device and inhabitant-inhabitant interactions. Instead of utility based quality-attribute preference, their work considers the services each inhabitant prefers and inhabitants are grouped into several classes where each in the same class has the same preference[28]. Pan et al. employ inverse reinforcement learning to analyze the preferences of taxi drivers when making decisions to look for passengers and study the dynamics of preferences of different groups of drivers over time [29]. Compared to their offline learning work indicating that human has unique preference changing over time, our work focus on preference online approximation and estimation based on the feedback on system behaviors from human.

The need to re-adjust utility function at run time based on changing user preferences has been raised [10]. Kakousis et al. define a cost function to enable the optimization of the quality attribute weights with the minimum variation in each optimization step based on user feedback [30]. However, their approach does not explicitly consider human complaints or corrections during optimization. user preference has also been incorporated into multi-objective genetic fuzzy rule selection for pattern classification problems and preference function is changeable according to the user's direct manipulation during evolution [31]. Wohlrab et al. use the predefined rule to adjust the weights based on the changing context or directly adjust the weight based on the input of human [9]. Sont et al. provide a simplified interface for users to directly revise the weights of existing goals after each round of adaptation so that the adaptation results will be closer to the user's preference [11]. Instead, we dynamically estimate human preference based on iteration algorithm on similarity analysis and human corrections so that the indicated strategy of updated preference comprises of the human expectation actions or averts human complaints.

Besides, Sukkerd et al. address the tradeoff rationale of multi-objective planning by explaining the strategy to end-users and then assess whether those decisions are in line with the user's goals and preference by conducting a human subject experiment [32]. Their explanations on how competing objectives make tradeoffs by contrasting their strategy to those of other rational alternatives with varying preferences could be adopted in our framework to improve users' confidence on system behaviors and decrease their intervention.

III. Preference Adaptation Framework

Preferences are usually associated with a high degree of uncertainty where each individual has unique preference changing over time. Considering the characteristic of users good at complaining when they are not satisfied, but not precisely specifying what they want, we propose a new self-adaptive framework considering the variation of human preference. Figure 2 shows a bird eye's view of our two-layer adaptation framework where the lower layer controller deals

with the environmental changes while the higher one updates preference based on the blurry input and behavior corrections from the user.

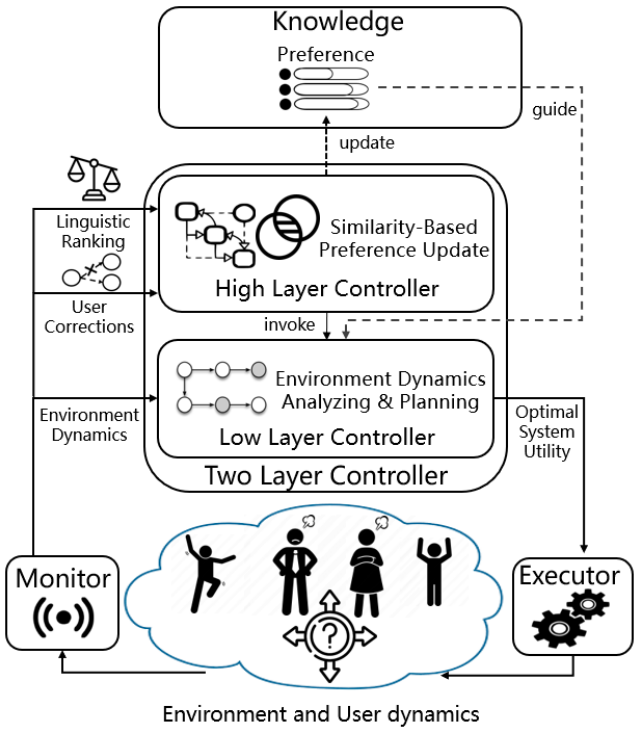


Fig. 2: Two-layer Adaptation Concerning Human Preference.

Monitor. Events generated in the environment indicating natural changes in the environmental factors or indicating preference changes from human are received. Typical example of natural events includes a new obstacle at a location on the map. There are two types of preference related dynamics: 1) direct input with new linguistic ranking from the user such as "efficiency is more important than security", and 2) indirect complaint with user corrections, e.g., refusing or disliking one certain step in a path. Monitor gathers and synthesizes these dynamics through the sensors deployed in the space. Then it classifies these information and distributes to two layers separately for the following decision making.

Low Layer Controller. During the environmental changes decision-making, this controller performs analysis based on the input of environmental dynamics from Monitor and checks whether violations or better satisfaction of goals exists. Then, an adaptation strategy aiming to counteract violations or better achieves goals defined is reasoned with the quality attribute aggregation function guided by the preference stored in the Knowledge. For each environmental adaptation situation, it generates a strategy if one exists as the adaptation to respond to unexpected changes, or prompts for a change in the design of the system if the violation cannot be handled. In this motivating scenario shown in Section II.A, the goal is to plan a strategy among all path candidates from starting points to destination and maximize the overall utility by trading off three quality attributes. In addition, this controller could be invoked on request of High Layer Controller for adaptation strategy simulation with the input of a potential preference condition and output of the optimal strategy with respect to this preference.

High Layer Controller. During preference update decision making, this High-Layer Controller receives user corrections from Monitor, indicating that the optimal strategy to the existing adaptation goal (i.e., aggregation utility function with preference stored in the Knowledge) does not cater to the user anymore and preference update is required. The update is based on the similarity analysis by iterating all possible preference candidates. For each preference candidate, this controller will invoke the Low Layer Controller for adaptation strategy simulation to obtain the candidate's strategy, which is required for similarity analysis and preference update. More details will be addressed at Section IV. In short, the preference candidate closest to the out-of-date and its indicated strategy conforming to human corrections with minimal differentiation from the previous strategy will be selected to update the preference at Knowledge. The basic idea behind this similarity analysis is that each individual changes his preference in a small step within a certain time frame. This is supported and consistent with those obtained by Pan et al. [29] who inversely learn from real data the preferences of taxi drivers whose magnitude of change is not large in a certain period of time. Besides, linguistic ranking is another possible direct input from users specifying the characterizations for two quality attributes: equally important; more important; much more important, etc. These characterizations can be reflected by rules for preference candidates checking, and this is useful in reducing the search space by directly dropping those candidates that do not suit for the desired ranking.

Executor. During execution, the strategy maximizing utility based on preference is enacted through actuators. Typical examples could be setting a new path from the starting point to the ending point and adaptively adjusting the moving directions towards the destination.

In this work, we focus on preference adaptation at run time, especially on dynamics of weights for the aggregation function, assuming the quality attributes and their corresponding utility functions each indicating its satisfaction level are settled. Environmental dynamics decision making based on preference and utility functions has been explored a lot and interested reader can refer to [33], [5]. Besides, we assume adequate monitoring in place including environment behavior and human corrections, as well as an execution environment through which selected adaptation strategies are enacted.

IV. Similarity-based Preference Adaptation

Run time decision making based on the trade-off of multiple quality attributes is not trivial because user preference is usually implicit and not easily elicited in a rigorous mathematical form. Besides, each person has unique preference changing over time and this factor builds as additional sources of uncertainty affecting the behavior of the self-adaptive system [34]. Thus, we introduce user preference (i.e., each

weight of quality attribute) as an explicit dynamic factor. Inspired by the finding from existing study [29], the key idea of our approach to enable automated reasoning about preference is based on similarity analysis by iterating all preference candidates and from the out-of-date to pick up the one with the great similarity and whose corresponding strategy conforms to human corrections. In this section, we present a general approach with more details on robot moving scenario, i.e., path planning problem solved by Reinforcement Learning (Sec IV.A), and formalize the distance (Sec IV.B) between preferences and between strategies in the form of path for similarity analysis. Then, an algorithm of updating preference will be given with possible linguistic ranking input (Sec IV.C).

A. Adaptation Strategy for a Preference Condition

In an adaptation decision making problem, the Low Layer Controller will reason about a strategy $str \in strs$ achieving the best possible utility among all possible strategy candidates under the aggregation utility function with a specific preference condition. A preference is formalized as a vector in which each element is the weight of a quality attribute $pre = \langle w_1, ..., w_n \rangle$. To this end, the strategy adaptation is defined by a general and abstract function, mapping the specific preference to its adaptation strategy:

$$\textsc{PreMapStr} : pre \rightarrow str \tag{1}$$

Strategy adaptation implementation for the Motivating Scenario. Strategy decision making in the robot scenario is the path planning problem, defined as Markov decision process (MDP). Given a preference $(w_1, ..., w_n)$ and a set of quality attribute functions $\{u_1, ..., u_n\}$, a preference applied MDP is defined as a 4-tuple $M \triangleq <S, A, P, R>$, where S is a set of states, A is a set of actions, $P : S \times A \times S \rightarrow [0, 1]$ is the state transition function. $R : S \times A \rightarrow \mathbb{R}$ is the quality attribute function where $r(s^t, a^t) = \sum_i w_i u_i(s^{t+1})$.

In this scenario, we consider the state transition function is determinate, meaning that given a state s there must exist a pair of state-action (a, s') satisfying $P(s, a, s') = 1$. Hence, the state transition function can be rewritten as $P : S \times A \rightarrow S$. The policy in the MDP is then defined as $\pi : S \rightarrow A$ and the optimal policy for the MDP (also the realization of function PREMAPSTR) is defined as follows:

$$\begin{aligned} \pi^* &= \arg\max_{\pi \in \Pi} \sum_{i=0}^{T} r(s^t, \pi(s^t)) \\ &= \arg\max_{\pi \in \Pi} \sum_{i=0}^{T} \sum_{i=1}^{n} w_i u_i(s^{t+1}) \\ &\text{s.t.,} \quad s^{t+1} = P(s^t, \pi(s^t)) \end{aligned} \tag{2}$$

where T is the terminal time step. Given an optimal policy π^*, the optimal path, i.e., adaptation strategy str, is defined as a state sequence $\langle s^1, s^2, ..., s^T \rangle$ satisfying $\forall t \in \{1, 2, ..., T - 1\}, s^{t+1} = P(s^t, \pi^*(s^t))$.

There are two kinds of methods, value-based and policy-based, for learning the optimal policy by Reinforcement Learning through accumulating the episodes (i.e., state, action, reward sequence) [35]. We adopt the value-based method by learning an optimal state-action value function (also known as Q function), i.e., $q(s, a) = \mathbb{E}[\sum_{k=0}^{T-t-1} \gamma^k R^{t+k+1} | s^t = s, a^t = a]$, where $R^t = r(s^t, a^t)$, and $\gamma \in [0, 1]$ is the discount factor. The optimal state-action value function can be learned using Temporal-Difference learning, i.e.,:

$$q(s, a) \leftarrow q(s, a) + \alpha[R + \gamma \max_{a \in A} q(s', a) - q(s, a)] \tag{3}$$

where $R = r(s, a)$, $s' = P(s, \pi(s))$,$\alpha \in [0, 1]$ is the learning rate and the optimal policy is computed as follows:

$$\pi^*(s) = \arg\max_{a \in A} q(s, a) \tag{4}$$

B. Similarity Between Preferences and Between Strategies

Similarity is also defined by a general, abstract function $\textsc{PreSim}(p_1, p_2)$ between a pair of preferences and $\textsc{StrSim}(str_1, str_2)$ between a pair of strategies. To measure the similarity between preferences p_1 and p_2 in n dimensions where $p_1 =< w'_1, w'_2, ..., w'_n >$ and $p_2 =< w''_1, w''_2, ..., w''_n >$, we use the cosine similarity CosSim of their weights vectors as the function implementation of PRESIM:

$$\begin{aligned} \textsc{PreSim}(p_1, p_2) &= \frac{p_1 \wedge p_2}{||p_1|| \times ||p_2||} \\ &= \frac{\sum_{i=1}^{n} w'_i \times w''_i}{\sqrt{\sum_{i=1}^{n} w'^2_i} \times \sqrt{\sum_{i=1}^{n} w''^2_i}} \end{aligned} \tag{5}$$

Strategy similarity implementation for the Motivating Scenario. For the robot, a strategy of a path is formalized as a sequence of states: $str = \langle s^1, s^2, ..., s^x \rangle$. However, the length of different paths are not the same, as shown in Schematic Figure 3. For instance, str_1 takes a detour with more states than str_2, str_3, or str_4. Thus, it is not appropriate to encode these strategies as vectors or use cosine similarity to measure the difference. Fortunately, the similarity measurement, i.e., implementation of function STRSIM, can be simplified by calculating the area between two path plans because every path has the same starting and ending point. Further consideration is to normalize the area differentiation in the range [0,1]. To this end, the similarity between a pair of paths str_1 and str_2 is implemented as follows:

$$\textsc{StrSim}(str_1, str_2) = \frac{\Delta_{Area}\{str_1, str_2\}}{\textsc{MaxArea}(\text{path space})} \tag{6}$$

where $\Delta_{Area}\{str_1, str_2\}$ is the area between path str_1 and str_2, i.e., the slash shadow area in Figure 3, while $MaxArea$ is the maximum area for the space of this path planning problem, i.e. the whole grey area as the path planning starts from the upper-left corner and ends at the bottom-right corner.

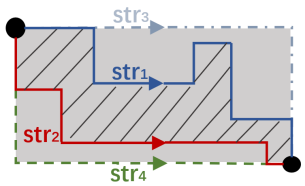


Fig. 3: Typical Paths Examples.

C. Preference Update Algorithm

Recall that the goal of preference update is to find a new preference candidate, i.e., a vector of quality attribute weights from the value space, closest to the outdated one and whose corresponding strategy conforms to human corrections. To this end, we propose a general algorithm to effectively output the new preference p_2 with its optimal strategy str_2 for this class of preference update problems. Algorithm 1 has as input an outdated preference p_1, a reasoned strategy str_1 achieving the highest utility based on p_1, as well as user corrections IC. Specifically, we consider IC as a set of states in the strategy being complained and rejected by the user in this motivating scenario.

The variable p_2 to be returned is initialized as well as its indicated optimal strategy s_2 calling the function PREMAPSTR by invoking the Low Layer Controller for adaptation strategy simulation. The algorithm then starts by iterating all possible values of n quality attributes at regular interval v pertaining that $w_1 + w_2 + ... + w_n = 1$ (line 3-6). The sampling interval determines the adaptation accuracy depending on the application scenarios and constraints of decision making latency. Obviously, the smaller the interval is, the larger search space and more number of candidates will be analyzed. Each preference candidate will be directly discarded if it does not satisfy the linguistic ranking from human input (lines 7-9). Then, each remaining candidate will be checked (line 11) if it is closer to the out-of-date p_1 by comparing the return value from function PRESIM, while its indicated strategy str_{tmp} (line 10) conforms to the corrections, i.e., not containing human complaining state $s^x \notin str_{tmp}$. If so, p_2 and str_2 are rewritten by the current candidate (lines 12-13). Typical example can be found in Figure 4 plot (a) where the preference candidate ($\langle 0.25, 0.2, 0.55 \rangle$) has minimal distance to the p_1 ($\langle 0.2, 0.3, 0.5 \rangle$) with similarity 0.98 from PRESIM and its indicated strategy averts human rejecting state in str_1. In addition, if the similarity of the current candidate with p_1 is equal to the so-far selected one and both of their strategies avert human complaints, the one whose strategy is nearer to the outdated strategy (i.e., $\text{STRSIM}(str_1, str_{tmp}) < \text{STRSIM}(str_1, str_2)$)) (line 15), will be re-selected (i.e., p_2 and str_2 are updated at lines 16-17), with the underlying idea to reuse existing strategy as much as possible. After iterating all the candidates, the algorithm returns the updated preference and strategy (line 23).

Linguistic ranking is the possible direct input from user specifying characterizations between quality attributes. For any two quality attribute QA_i and QA_j, the characterization can

Algorithm 1 The algorithm of Preference Update

Require: outdated preference $p_1 =< w_1^1, w_2^1, ..., w_n^1 >$, outdated adaptation strategy str_1 with user corrections $IC = (s^1, s^2, ..., s^m)$;
Ensure: updated preference p_2 with new strategy str_2;

```
1: random initialize p2
2: str2 = PREMAPSTR(p2)
3: for w1 in range(0,1,v) do
4:   for w2 in range (1 − w1,1,v) do
5:     ...
6:     for wn in range (1 − w1 − ... − wn−1,1,v) do
7:       if LINGUISTICRANKING(w1, ..., wn) == False
         then
8:         continue
9:       end if
10:      strtmp = PREMAPSTR(w1, ..., wn)
11:      if ( PRESIM(⟨w1, ..., wn⟩, p1)< PRESIM(p2, p1)
                                  &&
         ( ∀s^x ∈ IC : s^x ∉ strtmp ))
          then
12:        p2 ← ⟨w1, w2, ..., wn⟩
13:        str2 ← strtmp
14:      end if
15:      if ( PRESIM(⟨w1, ..., wn⟩, p1) = PRESIM(p2, p1)
                                  &&
         ( ∀s^x ∈ IC : s^x ∉ strtmp ∩
         STRSIM(str1, strtmp) < STRSIM(str1, str2)))
          then
16:        p2 ← ⟨w1, ..., wn⟩
17:        str2 ← strtmp
18:      end if
19:    end for
20:    ...
21:  end for
22: end for
23: return p2, str2
```

be reflected by encoding: 1) equally important $w_i = w_j$; more important $w_i > w_j$; much more important $w_i - w_j > threshold$; less important $w_i < w_j$; much less important $w_j - w_i > threshold$. These characterizations are useful in reducing the search space by directly dropping those candidates that do not suit for the desired ranking. For instance, a preference (safety, efficiency, privacy) candidate $\langle 0.2, 0.3, 0.5 \rangle$ does not comply with the characterization that "safety is more important than efficiency". Moreover, characterizations are time-sensitive and if there is no preference candidate satisfy both linguistic ranking and human corrections, indicating that characterizations are in conflict with user corrections, the latest corrections have a higher priority on preference update. This is because user preference is regarded as an explicit dynamic factor, which can not be simply hard-coded as static characterizations as most previous work does.

Algorithm Discussion. This preference update algorithm

with similarity analysis is general to any preference involved scenarios by customizing the strategy decision making and similarity between any forms of strategies (i.e., different implementation of function PREMAPSTR and STRSIM). In this motivating scenario, user corrections are abstracted and simplified as a set of states in the strategy denied by the user. Extensions to other form of corrections, such as rejected actions, expected actions or states, or even more complicated combinations, are straightforward by directly substituting the condition $\forall s^x \in IC : s^x \notin str_{tmp}$. In this work, we focus primarily on user preference is worthwhile to be updated at run time and provide one general update algorithm based on similarity analysis. Algorithm optimization or other decision making methods, such as strategy synthesis with user corrections can further be introduced if suitable for certain scenarios.

V. EVALUATION

To provide tool support for our preference adaptation framework, we realized a prototypical tool [1] based on robot good moving scenario in a 10×10 space considering three quality attributes: safety, efficiency, and privacy, described in Section II.A, with the iteration interval of 0.05. Thereupon, we evaluate our approach over three sets of experimentation; the experimental setup and results obtained are subsequently presented.

Our evaluation goals are three-fold; we seek to investigate **a) effectiveness.** We randomly choose four initial preference conditions and set up user corrections as one rejecting state on each adaptation strategy. This experiment illustrates the effectiveness of our algorithm to deal with preference update. **b) capability.** We set ground truth preference as user's internal preference and analyze how each preference condition, exploiting preference update algorithm and similarity analysis, approaches the ground truth preference with the input of user corrections, and what the preference update impact on the overall system utility results. This experiment entails the capability of our approach to enhance the utility and human satisfaction. **c) performance.** This experiment details the performance of our algorithm and how different linguistic ranking facilitates preference update by reducing the search space and decision making time.

A. Experimental Results : Effectiveness

Figure 4 (a-d) depicts the adaptation strategy (red arrow) with maximum utility for four preference conditions. We can observe that the robot chooses to take the middle path (i.e., red arrow), through the obstacles, when the quality attribute of safety holds only 0.2 shown in plot (a), while the upper middle path with comparably fewer obstacles is the choice at plot (b) when the proportion of safety increases to 0.3. Plot (c) illustrates the path passing through several private areas yielding better utility when safety with 0.4 is greater than privacy with 0.3. The strategy in plot (d) is the farthest path and detours to avoid any private areas or obstacles. This is due to very less proportion of efficiency with 0.1.

[1]https://github.com/sherry1912/PreferenceAdaptation.

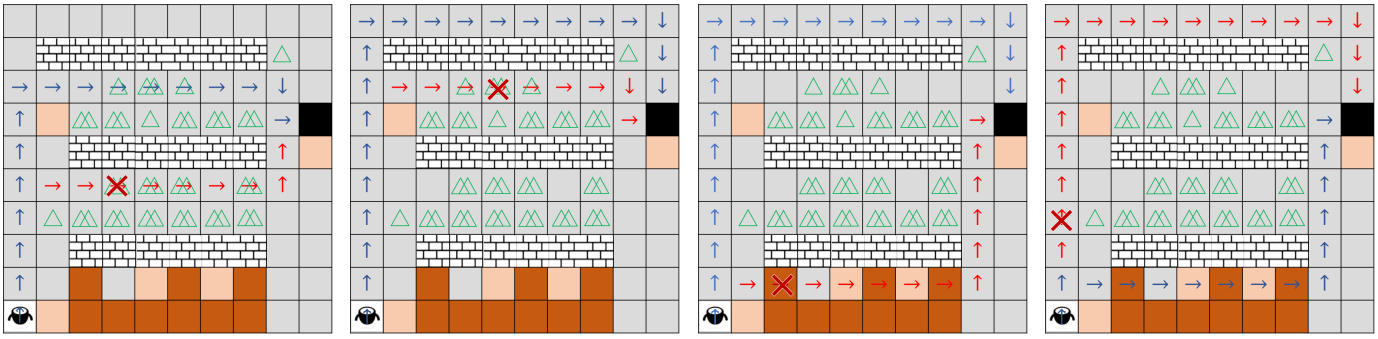
Figure 4 also depicts the updated preference with its new adaptation strategy (blue arrow) based on human correction, i.e., one denied state in the previous strategy. For the first case, as shown in plot (a), the new strategy changes to the upper path. This is because its updated preference ($\langle 0.25, 0.2, 0.55\rangle$) is closest to the outdated one ($\langle 0.2, 0.3, 0.5\rangle$) and the new strategy averts human complaining area. The increasing weight of safety in a small portion leads to the path with fewer obstacles. Plots (b-d) are analogous to (a), with generally small distance to the outdated preference conforming to human corrections.

However, one correction (i.e., states being complained) on the adaptation strategy by triggering the High Layer Controller for preference update does not necessarily approximate human real preference in practice without additional information. This is due to the assumption that each individual changes their preference in a small step not always true. Multiple times of corrections and preference updates might be required until there is no additional complaint from user anymore. One typical case that needs three times of update is the initial preference condition $\langle 0.05, 0.05, 0.90\rangle$ to ground truth preference $\langle 0.50, 0.40, 0.10\rangle$, as shown in Figure 5. Preference update with user correction happens when the current optimal strategy does not match the adaptation strategy of the ground truth with the first differentiation between two paths as the user complaining state. We can observe that the preference is closer to the ground truth after the first update but the similarity is only from 0.23 to 0.28, which is quite small due to the assumption behind our algorithm. The second and third updates explore the preference further afield with continuous corrections to the updated strategy. At the end, the preference reaches a fixed point $\langle 0.4, 0.3, 0.3\rangle$, though different from the ground truth with similarity 0.93, but these two preferences share the same strategy for this specific good moving task. Preference update can be further approximated to the ground truth with additional corrections when the task changes with other starting point and destination or environment changes, such as obstacle movement.

B. Experimental Results : Capability

We further statically analyze a discretized region of all possible state space with the interval of 0.05 and their similarity with the ground truth $\langle 0.5, 0.4, 0.1\rangle$ as shown in Figure 6. The x-axis and y-axis each represents w_2 and w_1. Each state is the initial preference condition and plot (a) presents the similarity results before preference update. Those states where w_1 is around 0 and w_2 around 0, mainly distributed in the lower left, have comparatively low similarity, but this can be gradually strengthened with preference update as shown from plot (b) to plot (d). After three corrections and updates, almost all of the states (i.e., initial preference conditions) have similarity greater than 0.8 with the ground truth preference, indicating this preference update framework and algorithm do support the self-adaptive system approaching user real preference.

Furthermore, we measure the number of preference update in need by settling each correction as one complaining state in



(a) ⟨0.2,0.3,0.5⟩→⟨0.25,0.2,0.55⟩ (b) ⟨0.3,0.2,0.5⟩→⟨0.35,0.15,0.5⟩ (c) ⟨0.4,0.3,0.3⟩→⟨0.4,0.25,0.35⟩ (d) ⟨0.45,0.1,0.45⟩→⟨0.5,0.2,0.3⟩

Fig. 4: Strategy for Updated Preference with User Corrections (preference = ⟨ Safety, Efficiency, Privacy⟩)

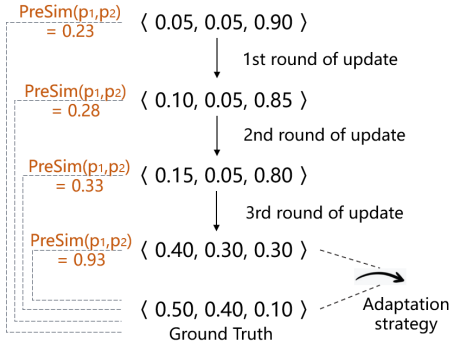


Fig. 5: An Illustration of Three Preference Updates.

the adaptation strategy, and the results are shown in Figure 7. Plot (a) presents the results for the ground truth $\langle 0.5, 0.4, 0.1\rangle$. This plot reveals that those states with comparative low similarity distributed in the lower left require three times of update to reach the fixed point due to the large distance. States requiring no update appears in the middle areas, when the distance is comparatively low with high similarity and their adaptation strategies are the same as the ground truth. These results are also explained by the fact that the greater the differentiation from user expectation (i.e., the ground truth), the more corrections human are about to give. Plot (b) is analogous to (a) but with states requiring 3 times mainly distributed in the upper left. This is because w_1 is near 1 in these areas, a little bit far away from the ground truth $\langle 0.1, 0.1, 0.8\rangle$.

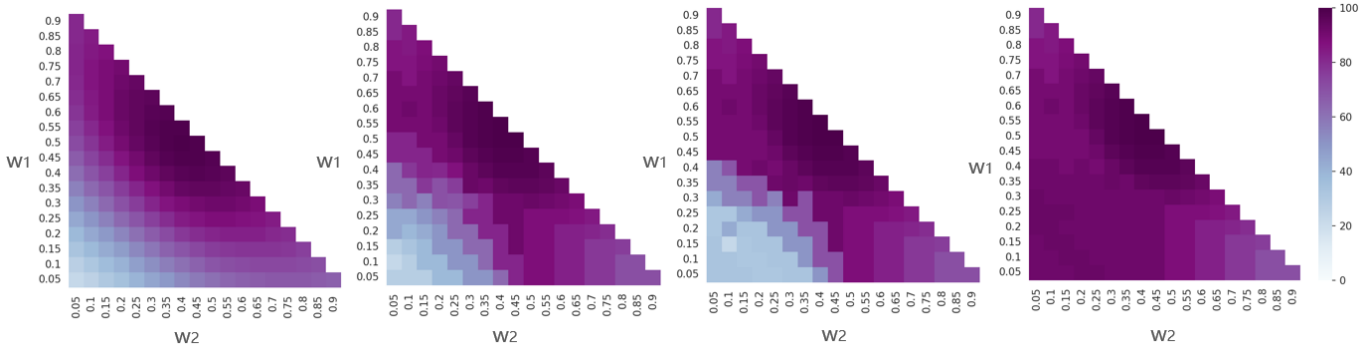
To clearly display the impact of preference update, We quantify the delta of utility between two strategies before and after the preference update in this set of experiments. We further calculate the ratio between the delta and the best possible utility for the ground truth to reduce the impact of the quality attribute function definition itself and illustrate the percentage variation.

$$ratio = \frac{U_{updatedstr,GT} - U_{outdatedstr,GT}}{U_{optimalstr,GT}}$$

Figure 9 depict the impact of preference update (i.e., ratio) by employing the fixed values for human real preference (i.e., ground truth $\langle 0.1, 0.1, 0.8\rangle$), and analyze a discretized region of all possible states. In the first round, as shown in plot (a), the update results are generally positive. This can be observed in the red upward area where the ratio is greater than 0 (i.e., utility delta¿0), up to about 0.35. However, there is a small proportion of preference conditions (14 in total of 171) that one preference update might lead to negative results (i.e., ratio < 0) as shown in the downward area with blue color. A typical initial preference condition example is $\langle 0.2, 0.1, 0.7\rangle$ and it will be updated to $\langle 0.2, 0.05, 0.75\rangle$, farther away to the ground truth. This is due to the algorithm logic that the preference candidate closest to the previous one and it indicated strategy not stepping on human complaints will be selected, which does not always work well. However, the results of these conditions will gradually get better with the continuous update by inputting more information and corrections from user. This is illustrated as the number of states in negative ratio decreases to 0 in plot (c). Figure 8 shows the utility ratio results for the ground truth $\langle 0.1, 0.1, 0.8\rangle$. In general, there are small number of preference conditions leading to negative results in both plot (a) and (b). After three preference updates, the ratio for every state is greater or equal to 0, with 109 out of 171 even greater than 0.33, indicating that preference update enhance utility by roughly one-third.

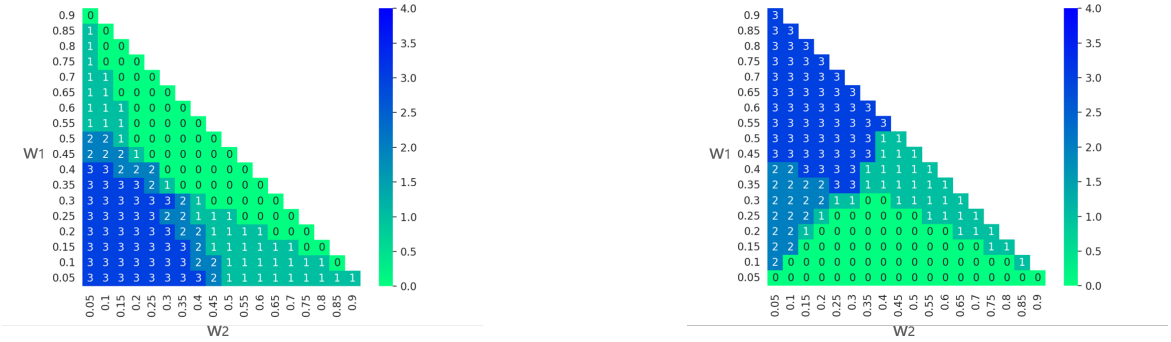
C. Experimental Results : Performance

The third set of experimentation illustrates the impact of linguistic ranking as additional input, as shown in Figure 10. The overall setting is similar to experimentation in Figure 7. For plot (a), user informed their expected characterizations of quality attributes: 1) security equal to efficiency, and 2) privacy much more important than security. Reasoning about adaptation strategy for one preference candidate by reinforcement learning technique for this robot scenario takes around 50 episodes, about 0.2s, to convergence. Without linguistic ranking, the experimentation time frame for one preference update takes around 35s due to the iteration of all potential preference candidates (i.e., 171 in total with interval 0.05) as well as their indicated strategy. It can be reduced to 1s with linguistic ranking by directly discarding majority of candidates at iteration that do not meet the characterizations between quality attributes. Besides, those preference conditions requir-



(a) Initial Preference Condition (b) After One Round of Update (c) After Two Rounds of Update (d) After Three Rounds of Update

Fig. 6: Similarity Analysis for Ground Truth Preference $\langle 0.5, 0.4, 0.1 \rangle$.



(a) ground_truth_preference = $\langle$0.5,0.4,0.1$\rangle$ (b) ground_truth_preference = $\langle$0.1,0.1,0.8$\rangle$

Fig. 7: Number of Preference Update in Need

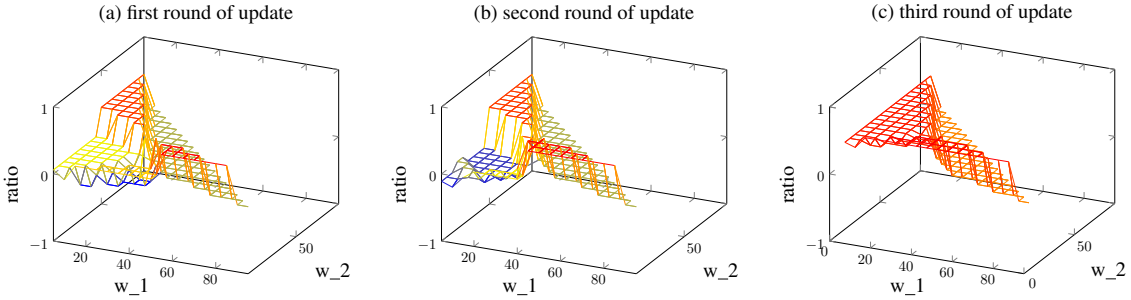


Fig. 8: Analysis Results for Ground Truth $\langle 0.5, 0.4, 0.1 \rangle$

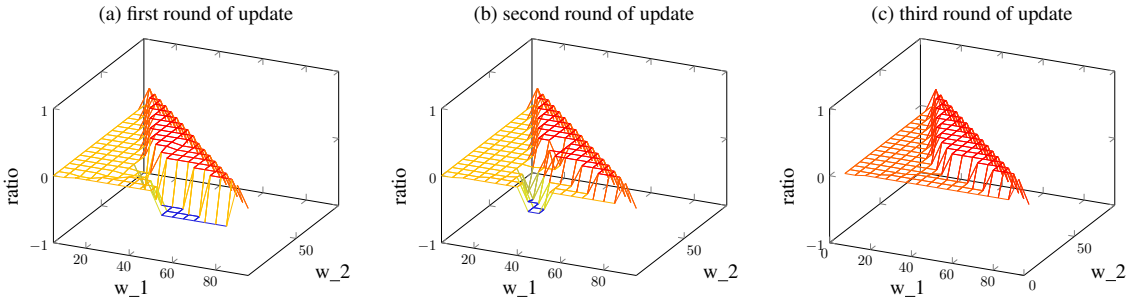
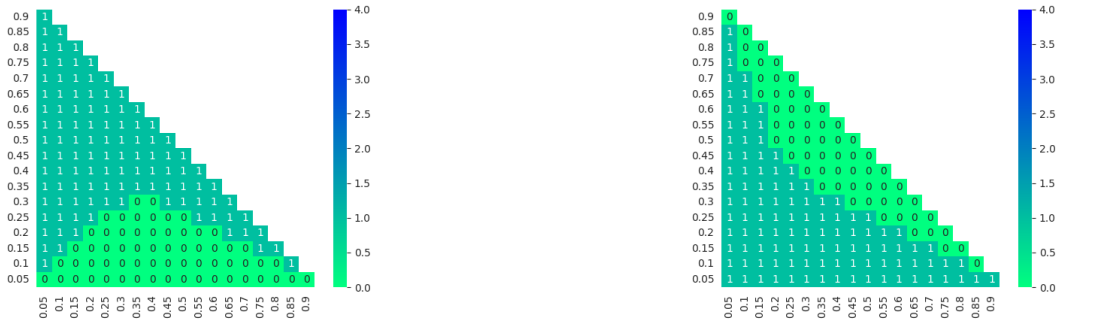


Fig. 9: Analysis Results for Ground Truth $\langle 0.1, 0.1, 0.8 \rangle$



(a) ground_truth_preference = ⟨0.1,0.1,0.8⟩ (b) ground_truth_preference = ⟨0.5,0.4,0.1⟩

Fig. 10: Number of Preference Update in Need with Linguistic Ranking

ing two or three times of update as shown in Figure 7 just need one update to the fixed point because user's input of linguistic ranking reduces the search space and helps our algorithm locates and approaches user real preference in a smaller number of trial. Plot (b) shows the analogous results by noticing that: 1) security more important than efficiency, and 2) efficiency more important than security.

In summary, the results of analyzing robot good-moving scenario have shown that: (i) user preference can be approached in our proposed framework even if user cannot accurately express their preferences in a mathematical form, (ii) the approximation process may not always be smooth as the first few updates may even cause utility degradation (i.e., ratio < 0), however, it will eventually lead to the decreasing distance to user real preference with increasing similarity by their feedback and corrections on the adaptation strategy, (iii) preference update can enhance the overall utility (i.e., ratio > 0) in most cases and there is ultimately no reduction in utility even in some worst cases (i.e., ratio $= 0$), (iv) linguistic ranking from user input generally facilitates the approximation process, (i.e., decreasing number of corrections and preference updates in need) and accelerate the update speed with shrinking search space by discarding those states not conforming to the linguistic ranking.

VI. Conclusions and Discussion

User preference with uniqueness and dynamic nature has been considered as one essential uncertainty to design self-adaptive systems. To this end, we presented a two-layer general framework to 1) update the preference based on similarity analysis and resolution of human unsatisfaction, and 2) reason about adaptation strategy that best matches the updated preference and achieves the optimal utility under uncertain environment.

One limitation of our initial evaluation of this work was based on a theoretical robot good-moving case study, which has illustrated the potential of this approach. Although we did not directly correlate our analytical results with actual systems through an empirical study, our findings are partially supported by those obtained by Pan et al. [29], whose results illustrate that self-improving drivers tend to keep adjusting their preferences to increase earning efficiency based on collected data from DiDi Chuxing Inc (Uber-like company at China). Our future research is also planned in order to evaluate in the context of comprehensive self-driving scenarios, using actual devices and real user feedback.

The second limitation of our approach involves the iteration of all potential preference candidates with their indicated strategy, inducing a high overhead which may be not bearable at run time, especially for those reasoning methods such as reinforcement learning or Markov Decision Process. To address this issue, hybrid planning [36] or plan reuse [37] might be leveraged to reduce the cost of preference planning. Knowledge base can store the pre-computed plans so that the controller could potentially reuse to react as quickly as needed. Besides, if the preference reasoning is only allowed during a limited time frame, the interval of iteration could be relaxed in need. Another limitation is the assumption behind the update algorithm and reinforcement learning that a certain preference candidate corresponds to only one optimal adaptation strategy, which may not be true in practice. In these cases, preference renewal could be deferred until none of its indicated strategies meet user complaints and corrections.

Our initial investigation suggests a number of additional research directions, such as considering constraints (including hard constraints and soft constraints) beyond user preference in the goal model and conflicts between different types of constraints and between hard constraints with user complaints; adopting existing methods such as fuzzy preference relations with induced orders [26], or preference elicitation and negotiation [26] to align with user preference offline before deployment, minimizing the number of preference updates; and finally, explaining the adaptation strategy to user [32] by how multiple quality attributes compete each other to improve user confidence on adaptive system so to make appropriate corrections in need. This framework could be further adjusted and extended in any human-involved systems with preference by customizing the part of similarity analysis between strategies in the algorithm. Scenarios with strategy specified in tactic language stitch [38] can also be considered in future work.

REFERENCES

[1] R. U. Bilsel, G. Büyüközkan, and D. Ruan, "A fuzzy preference-ranking model for a quality evaluation of hospital web sites," *Int. J. Intell. Syst.*, vol. 21, no. 11, pp. 1181–1197, 2006.

[2] J. Kephart, "Viewing autonomic computing through the lens of embodied artificial intelligence: A self-debate." *Keynote at the 16th Symposium on Software Engineering for Adaptive and Self-Managing Systems. (SEAMS 2021)*, 2021.

[3] N. Li, J. Cámara, D. Garlan, B. R. Schmerl, and Z. Jin, "Hey! preparing humans to do tasks in self-adaptive systems," in *16th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS@ICSE 2021, Madrid, Spain, May 18-24, 2021*. IEEE, 2021, pp. 48–58.

[4] N. Li, M. Zhang, E. Kang, and D. Garlan, "Engineering secure self-adaptive systems with bayesian games," in *Fundamental Approaches to Software Engineering - 24th International Conference, FASE 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings*, ser. Lecture Notes in Computer Science, E. Guerra and M. Stoelinga, Eds., vol. 12649. Springer, 2021, pp. 130–151.

[5] J. Cámara, G. A. Moreno, D. Garlan, and B. Schmerl, "Chapter 7 - evaluating trade-offs of human involvement in self-adaptive systems," in *Managing Trade-Offs in Adaptable Software Architectures*, I. Mistrik, N. Ali, R. Kazman, J. Grundy, and B. Schmerl, Eds. Boston: Morgan Kaufmann, 2017, pp. 155 – 180.

[6] N. Li, J. Cámara, D. Garlan, and B. R. Schmerl, "Reasoning about when to provide explanation for human-involved self-adaptive systems," in *IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2020, Washington, DC, USA, August 17-21, 2020*, 2020, pp. 195–204.

[7] R. Sukkerd, "Improving transparency and understandability of multi-objective probabilistic planning," *Thesis Proposal – School of Computer Science Institute for Software Research Software Engineering, Carnegie Mellon University*, pp. 1–41, 2018.

[8] N. Li, S. Adepu, E. Kang, and D. Garlan, "Explanations for human-on-the-loop: A probabilistic model checking approach," in *Proceedings of the 15th International Symposium on Software Engineering for Adaptive and Self-managing Systems (SEAMS)*, 2020, to appear.

[9] R. Wohlrab, R. Meira-Góes, and M. Vierhauser, "Run-time adaptation of quality attributes for automated planning," in *International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2022, Pittsburgh, PA, USA, May 22-24, 2022*, B. R. Schmerl, M. Maggio, and J. Cámara, Eds. ACM/IEEE, 2022, pp. 98–105.

[10] V. Lesch, M. Hadry, S. Kounev, and C. Krupitzer, "Utility-based vehicle routing integrating user preferences," in *19th IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events, PerCom Workshops 2021, Kassel, Germany, March 22-26, 2021*. IEEE, 2021, pp. 263–268.

[11] H. Song, S. Barrett, A. Clarke, and S. Clarke, "Self-adaptation with end-user preferences: Using run-time models and constraint solving," in *Model-Driven Engineering Languages and Systems - 16th International Conference, MODELS 2013, Miami, FL, USA, September 29 - October 4, 2013. Proceedings*, ser. Lecture Notes in Computer Science, A. Moreira, B. Schätz, J. Gray, A. Vallecillo, and P. J. Clarke, Eds., vol. 8107. Springer, 2013, pp. 555–571.

[12] G. Karagiannakis, A. Baccaglini-Frank, and Y. Papadatos, "Mathematical learning difficulties subtypes classification," *Frontiers in Human Neuroscience*, vol. 8, 01 2014.

[13] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Eds., *Software Engineering for Self-Adaptive Systems [outcome of a Dagstuhl Seminar]*, ser. Lecture Notes in Computer Science, vol. 5525. Springer, 2009.

[14] C. A. N. Soules and G. R. Ganger, "Why can't I find my files? new methods for automating attribute assignment," in *Proceedings of HotOS'03: 9th Workshop on Hot Topics in Operating Systems, May 18-21, 2003, Lihue (Kauai), Hawaii, USA*, M. B. Jones, Ed. USENIX, 2003, pp. 115–120.

[15] J. Cámara, G. A. Moreno, and D. Garlan, "Reasoning about human participation in self-adaptive systems," in *10th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS, Florence, Italy, May 18-19, 2015*, 2015, pp. 146–156.

[16] R. Sukkerd, D. Garlan, and R. G. Simmons, "Task planning of cyber-human systems," in *Software Engineering and Formal Methods - 13th International Conference, SEFM 2015, York, UK, September 7-11, 2015. Proceedings*, 2015, pp. 293–309.

[17] E. Lloyd, S. Huang, and E. Tognoli, "Improving human-in-the-loop adaptive systems using brain-computer interaction," in *12th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS@ICSE 2017, Buenos Aires, Argentina, May 22-23, 2017*, 2017, pp. 163–174.

[18] J. E. Fischer, C. Greenhalgh, W. Jiang, S. D. Ramchurn, F. Wu, and T. Rodden, "In-the-loop or on-the-loop? interactional arrangements to support team coordination with a planning agent," *Concurrency and Computation: Practice and Experience*, pp. 1–16, 2017.

[19] H. Goldsby, P. Sawyer, N. Bencomo, B. H. C. Cheng, and D. Hughes, "Goal-based modeling of dynamically adaptive system requirements," in *15th Annual IEEE International Conference and Workshop on Engineering of Computer Based Systems (ECBS 2008), 31 March - 4 April 2008, Belfast, Northern Ireland*. IEEE Computer Society, 2008, pp. 36–45.

[20] J. Kramer and J. Magee, "Self-managed systems: an architectural challenge," in *International Conference on Software Engineering, ISCE 2007, Workshop on the Future of Software Engineering, FOSE 2007, May 23-25, 2007, Minneapolis, MN, USA*, L. C. Briand and A. L. Wolf, Eds. IEEE Computer Society, 2007, pp. 259–268.

[21] M. Salehie and L. Tahvildari, "A quality-driven approach to enable decision-making in self-adaptive software," in *29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, USA, May 20-26, 2007, Companion Volume*. IEEE Computer Society, 2007, pp. 103–104.

[22] ——, "A weighted voting mechanism for action selection problem in self-adaptive software," in *Proceedings of the First International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2007, Boston, MA, USA, July 9-11, 2007*. IEEE Computer Society, 2007, pp. 328–331.

[23] J. Karlsson and K. Ryan, "A cost-value approach for prioritizing requirements," *IEEE Softw.*, vol. 14, no. 5, pp. 67–74, 1997.

[24] C. Ghezzi and A. M. Sharifloo, "Dealing with non-functional requirements for adaptive systems via dynamic software product-lines," in *Software Engineering for Self-Adaptive Systems II - International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*, ser. Lecture Notes in Computer Science, R. de Lemos, H. Giese, H. A. Müller, and M. Shaw, Eds., vol. 7475. Springer, 2010, pp. 191–213.

[25] N. Mehdipour, C. I. Vasile, and C. Belta, "Specifying user preferences using weighted signal temporal logic," *IEEE Control. Syst. Lett.*, vol. 5, no. 6, pp. 2006–2011, 2021.

[26] I. C. Parmee and D. Cvetkovic, "Preferences and their application in evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 42–57, 2002.

[27] R. Wohlrab and D. Garlan, "Defining utility functions for multi-stakeholder self-adaptive systems," in *Requirements Engineering: Foundation for Software Quality - 27th International Working Conference, REFSQ 2021, Essen, Germany, April 12-15, 2021, Proceedings*, ser. Lecture Notes in Computer Science, F. Dalpiaz and P. Spoletini, Eds., vol. 12685. Springer, 2021, pp. 116–122.

[28] Z. Lin and L. Fu, "Multi-user preference model and service provision in a smart home environment," in *IEEE Conference on Automation Science and Engineering, CASE 2007, September 22-25, 2007. Scottsdale, Arizona, USA*. IEEE, 2007, pp. 759–764.

[29] M. Pan, W. Huang, Y. Li, X. Zhou, Z. Liu, R. Song, H. Lu, Z. Tian, and J. Luo, "DHPA: dynamic human preference analytics framework: A case study on taxi drivers' learning curve analysis," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 1, pp. 8:1–8:19, 2020.

[30] K. Kakousis, N. Paspallis, and G. A. Papadopoulos, "Optimizing the utility function-based self-adaptive behavior of context-aware systems using user feedback," in *On the Move to Meaningful Internet Systems: OTM 2008, OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008, Monterrey, Mexico, November 9-14, 2008, Proceedings, Part I*, ser. Lecture Notes in Computer Science, R. Meersman and Z. Tari, Eds., vol. 5331. Springer, 2008, pp. 657–674.

[31] Y. Nojima and H. Ishibuchi, "Incorporation of user preference into multi-objective genetic fuzzy rule selection for pattern classification problems," *Artif. Life Robotics*, vol. 14, no. 3, pp. 418–421, 2009.

[32] R. Sukkerd, R. G. Simmons, and D. Garlan, "Tradeoff-focused contrastive explanation for MDP planning," in *29th IEEE International Conference on Robot and Human Interactive Communication, RO-MAN*

2020, Naples, Italy, August 31 - September 4, 2020. IEEE, 2020, pp. 1041–1048.

[33] S. Adepu and A. Mathur, "Assessing the effectiveness of attack detection at a hackfest on industrial control systems," *IEEE Transactions on Sustainable Computing*, 2018.

[34] N. Esfahani and S. Malek, "Uncertainty in self-adaptive software systems," in *Software Engineering for Self-Adaptive Systems II - International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers*, 2010, pp. 214–238.

[35] C. F. Hayes, R. Radulescu, E. Bargiacchi, J. Källström, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz, E. Howley, A. A. Irissappane, P. Mannion, A. Nowé, G. de Oliveira Ramos, M. Restelli, P. Vamplew, and D. M. Roijers, "A practical guide to multi-objective reinforcement learning and planning," *Auton. Agents Multi Agent Syst.*, vol. 36, no. 1, p. 26, 2022.

[36] A. Pandey, I. Ruchkin, B. R. Schmerl, and D. Garlan, "Hybrid planning using learning and model checking for autonomous systems," in *IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2020, Washington, DC, USA, August 17-21, 2020*. IEEE, 2020, pp. 55–64.

[37] C. Kinneer, D. Garlan, and C. L. Goues, "Information reuse and stochastic search: Managing uncertainty in self-* systems," *ACM Trans. Auton. Adapt. Syst.*, vol. 15, no. 1, pp. 3:1–3:36, 2021.

[38] S. Cheng and D. Garlan, "Stitch: A language for architecture-based self-adaptation," *J. Syst. Softw.*, vol. 85, no. 12, pp. 2860–2875, 2012.