To Ask or Not to Ask: A User Annoyance Aware Preference Elicitation Framework for Social Robots

Balint Gucsi, Danesh S. Tarapore
University of Southampton, UK
{bg1u17,d.s.tarapore}@soton.ac.uk
Christopher Amato
Northeastern University, USA
c.amato@northeastern.edu

William Yeoh
Washington University in St. Louis, USA
wyeoh@wustl.edu
Long Tran-Thanh
University of Warwick, UK
long.tran-thanh@warwick.ac.uk

Abstract— **In this paper we investigate how social robots can efficiently gather user preferences without exceeding the allowed user annoyance threshold. To do so, we use a Gazebo based simulated office environment with a TIAGo Steel robot. We then formulate the user annoyance aware preference elicitation problem as a combination of tensor completion and knapsack problems. We then test our approach on the aforementioned simulated environment and demonstrate that it can accurately estimate user preferences.**

I. Introduction

With the rise of artificial intelligence, autonomous and intelligent social robots (e.g., assistive robots [1], or interactive collaborative robots [2]), are becoming more and more ubiquitous in our everyday life. These robots typically focus on identifying the preferences of human users in order to efficiently execute collaborative/assistive social tasks, ranging from making coffee at the right time, helping workers in packing, to assisting elderly persons in their everyday life. To date, such human-robot collaboration is achieved by using machine learning techniques for the robot to learn user preferences through a prolonged observation period [3], [4]. However, from the user's perspective, this is usually not acceptable as these learning models typically require a significant amount of observed data in order to achieve good accuracy [5]. This may lead to user dissatisfaction in many cases (as the learning time would take too long), resulting in the user losing trust and interest in collaborating/interacting with the robot [6].

To overcome this issue, interactive preference elicitation has been proposed as a technique to speed up the learning process (i.e., to proactively ask the user to provide information about their preferences) [7], [8], [9], [10]. This is typically done through some sort of human-robot interaction or feedback framework. This approach, however, comes with the cost of user annoyance. In particular, by answering too many elicitation questions, the users may feel upset and bothered, which may also discourage the user from interacting/collaborating with the robot in the future [6], [11].

In this work, we ask the following research question: what would be an efficient way to gather information about the user's preferences without annoying them? Put differently, what type of questions should a robot ask, and when should it stop asking before it completely annoys the user, and can ask no more questions? To answer these questions, we design a preference elicitation framework for collaborative robots in office environments, that takes into account user annoyance through a repeated elicitation process as follows: The robot formulates the user preference query process as a tensor completion problem, for which it uses heuristics taken from the knapsack optimisation literature to identify which questions to ask. Each question has an associated annoyance cost, and the robot stops asking questions once the annoyance cost budget of the users has been exceeded. We also investigate the case when faulty task executions cause a decrease in future user preference values (e.g., if the robot fails to complete a task, the user might lose interest in asking the robot to do the same task in the future).

To demonstrate the efficiency of our model, we build a simulator based office environment in which a TIAGo Steel robot[1] collaborates with human users by executing a set of tasks for them. Our experiments show that our heuristics can recover the user preferences with high accuracy by asking as few as $3-6$ questions per user.

II. Related Work

Within the AI/robotics community, there are a number of work addressing preference elicitation problems with incomplete information. Many of the work focus on learning the preferences of the users [3], [4], [12]. However, these techniques cannot actively learn the unobserved preferences. On the other hand, interactive preference elicitation models can gather full preference information by interacting with the user, but without taking into account user annoyance during the process [8], [9], [10]. Notable exceptions are the work of [5], [13], that integrate user annoyance costs into the elicitation process. However, they do not deal with annoyance caused by physical execution failures, and thus, cannot be employed to the social robotics setting.

In addition, in the automated negotiation literature from the multiagent systems community, there exist some work in which queries to ask users can be associated with arbitrary annoyance costs [14], [15]. However, in these studies, the annoyance costs are typically included in the objective function. Thus, the proposed solutions typically select a query that ensures the highest expected negotiation payoff (i.e., a

Corresponding author: Long Tran-Thanh, University of Warwick.

[1]`https://tiago.pal-robotics.com`

combination of utility and annoyance cost). As such, it is regarded as a one-shot optimization problem. However, this approach cannot adapt to changes in user preferences, which often occur in many real-world domains. In contrast, to achieve an adaptive, sustainable sequence of human-robot interactions our proposed framework selects the question to ask based on the current state of partially filled preference tensor and the remaining predefined annoyance cost budget. Thus, we are more interested in an optimal sequence of queries whose total annoyance cost does not exceed a budget.

III. Preference Elicitation Model

In this section, we formulate the process a social robot needs to perform in order to learn the preferences of their human collaborators on executing certain tasks. In particular, this preference elicitation problem is defined formally as:

Definition 1: A *preference elicitation problem* is a tuple $\langle A, T, U, \mathscr{Q}, B(\cdot), \mathscr{B}\rangle$, where

- $A = \{1, 2, \dots, |A|\}$ is the set of tasks, $T = \{1, 2, \dots, |T|\}$ is the set of discrete time slots, and $U = \{1, 2, \dots, |U|\}$ is the set of users. Our goal is to elicit the preference of each user k about executing task i at time slot j.
- $\mathscr{Q}$ is a finite set of questions that can be asked.
- $B(\cdot) : 2^{\mathscr{Q}} \to \mathbb{R}^+$ is the annoyance cost function that takes as input a sequence of questions and determines the annoyance cost of asking those questions.
- $\mathscr{B} > 0$ is the annoyance cost budget (i.e., the maximum amount of annoyance cost that can be incurred). For now we assume that this budget is the same for each user k, but note that our solution can also be easily adopted to heterogeneous budgets.

Preference Tensor: The preferences of a user for having task A scheduled in T is modeled as a 3-dimensional tensor M, called *preference tensor*, where each row corresponds to a task $a \in A$, each column corresponds to a time slot $t \in T$, and the third dimension represents human users.[2] We use $p_k(i, j) = M_{i,j,k}$ to denote the preference of the user k having task $i \in A$ scheduled at time slot $j \in T$, where the greater the value of $p_k(i, j)$, the more likely the user will want task i to be carried out at time slot j. For simplicity and w.l.o.g. we assume that these preferences are integers in $\{1, \dots, 10\}$.

In our problem, it is reasonable to assume that tasks that need to be scheduled and executed are interdependent across tasks, users, and their execution times (e.g., see [16]). For example, consider a particular user and their corresponding rows (i.e., dimension A) in M that represents the execution preference of the tasks. The linear dependency between two rows a_1 and a_2 of M implies that the user has similar preferences on a_1 and a_2 regarding their execution time. Similarly, the correlation between users k_1 and k_2 can be captured by the dependency between their corresponding task-time matrices in M. This implies that the *preference tensor M is approximately low rank*, which allows us to perform exact tensor completion via convex optimization to recover the preference tensor.

[2]It is worth noting that the periodicity of the task execution's preferences can also be represented by preference tensors and, thus, can be handled by our approach.

Preference Elicitation Questions: Each question $q \in \mathscr{Q}$ is associated with:

- a *cognitive annoyance cost* $c(q) > 0$ (i.e., cognitive effort required of the user to answer the question).
- a set of entries $f(q) \subseteq \{(i, j, k) \mid i \in A, j \in T, k \in U\}$ and a set of values $\{p_k^q(i, j) \mid (i, j, k) \in f(q)\}$.

Intuitively, given a user k's response to a question $q \in \mathscr{Q}$, the robot fills a preference value $p_k^q(i, j)$ into the entry (i, j, k) of the preference tensor M (i.e., $M_{i,j,k} = p_k^q(i, j)$) for each element of $f(q)$. We assume that users are honest in that when two different questions fill some common entries, the preference values are consistent, i.e., $\forall q_1, q_2, (i, j, k) \in f(q_1) \cap f(q_2) : p_k^{q_1}(i, j) = p_k^{q_2}(i, j)$. Therefore, for convenience, we omit the superscript q in $p_k^q(i, j)$ from here on.

Example 1: Consider the following two preference elicitation questions q_1 and q_2:

- q_1: How likely you would like to have the bins emptied between 12-2pm?
- q_2: How likely you would leave the window #1 opened between 4-8pm?

where we use the following scale to measure the preference:

HIGHLY UNLIKELY									EXTREMELY LIKELY
1	2	3	4	5	6	7	8	9	10

Now assume that $T = \{1, \dots, 12\}$ where each time slot represents an interval of 2 hrs slot, we have $f(q_1) = \{(2, 7, 1)\}$ and $f(q_2) = \{(1, 9, 2), (1, 10, 2)\}$ if the robot sent q_1 to user 1, and q_2 to user 2, respectively. In addition, a possible annoyance cost function is one that assigns $c(q_1) = 2$ and $c(q_2) = 3$, as a user is likely to be less bothered answering a single question that asks the preferences over two time steps than two questions that each asks the preference for a single time step. Hence, $c(q_2) \leq 2c(q_1)$.

Given a question $q \in \mathscr{Q}$ and preference tensor M, $M(q)$ is *the update of M by q* based on the user's response to q and is defined as $M(q)_{i,j,k} = p_k(i, j)$ *if* $(i, j, k) \in f(q)$ and $M(q)_{i,j,k} = M_{i,j,k}$ *otherwise*. For a sequence of questions $Q = \langle q_1, q_2, \dots, q_\ell\rangle$, let $f(Q) = \bigcup_{q \in Q} f(q)$ and $M_1 = M(q_1)$, $M_2 = M_1(q_2)$, ..., $M(Q) = M_\ell = M_{\ell-1}(q_\ell)$, i.e., $M(Q)$ is the result of recursively updating M by questions q_1, q_2, ..., q_ℓ. It is easy to see that the order of questions by which M is updated does not affect the resulting tensor.

Annoyance Cost Model: Our proposed framework is generic and therefore it is compatible with any annoyance cost model. However, in this paper, we use the model proposed by [17] since *(i)* it has been used in a large body of literature (e.g., [18], [19]), and *(ii)* it also gives the annoyance cost function with more exponential and logarithmic appearances for more unwilling and willing users, respectively, that fits the users' typical behaviour in office environments [5].

Let $Q \subseteq \mathscr{Q}$ be a sequence of questions that has been asked thus far. The annoyance cost model by [17] defines "*annoyance cost so far*" (ASF) as:

$$ASF_Q = \sum_{q \in Q} c(q)\beta^{e(q)} \qquad (1)$$

where $0 < \beta \leq 1$ is a discount factor used to represent the diminishing impact of interactions over time (i.e., questions asked long ago will be less bothersome than questions recently asked) and $e(q)$ is the amount of elapsed time since q was asked. The total *annoyance cost* is then computed as:

$$C(Q) = Init + \frac{1-\alpha^{ASF_Q}}{1-\alpha} \quad (2)$$

where[3] $\alpha = 1.26 - 0.05w$, $Init = 10 - w$, and w denotes the *willingness* of a user to interact (i.e., answering questions) on a scale of 0 (for unwilling users) to 10 (for willing users).

In our application, we assume that the robot will ask questions consecutively and will thus use the number of questions that has been asked after asking q as the value for $e(q)$. For example, assume the sequence of questions $Q = \langle q_1, q_2, q_3 \rangle$, then $e(q_1) = 2$, $e(q_2) = 1$, and $e(q_3) = 0$. With $\beta = 0.95$ and $c(q_i) = 1$ for $1 \le i \le 3$, we have $AFS_Q = 1 + 0.95^1 + 0.95^2 = 2.8525$.

Objective Function: Let M be a preference tensor, where all entries are initially set to *null* and $M(Q)$ be the update of M by a sequence of questions $Q \subseteq \mathscr{Q}$. Our goal is to estimate the *null* entries of $M(Q)$ using the non-*null* entries in $M(Q)$ using standard tensor completion techniques from the literature (see, e.g., [20], [21], [22]). Given a tensor completion algorithm $\mathscr{L}$, let $\widehat{M^{\mathscr{L}}(Q)}$ be resulting estimated tensor by $\mathscr{L}$ with input $M(Q)$. When $\mathscr{L}$ is unspecified or clear from the context, we will omit it from the superscript.

Finally, the goal of the problem is to identify an optimal sequence of questions Q^* from $\mathscr{Q}$ and tensor completion algorithm $\mathscr{L}^*$:

$$\langle Q^*, \mathscr{L}^* \rangle = \arg\min_{Q,\mathscr{L}} \frac{\overbrace{||\mathscr{M} - \widehat{M^{\mathscr{L}}(Q)}||_1}^{W(Q,\mathscr{L})}}{\underbrace{|A| \times |T| - |f(Q)|}_{Z(Q)}} \text{ s.t. } C(Q) \le \mathscr{B} \quad (3)$$

where $\mathscr{B}$ is the annoyance cost budget; $\mathscr{M}$ is the true (oracle) preference tensor that can be achieved in the ideal scenario where preferences for *all* tasks and users at every time slot should be elicited; and $||X||_1$ is the L_1 norm (i.e., sum of all absolute values of entries) of tensor X. In Equation (3), $W(Q,\mathscr{L})$ denotes the differences between $\mathscr{M}$ and $\widehat{M^{\mathscr{L}}(Q)}$, and $Z(Q)$ is the number of unfilled entries in $M(Q)$.

Solution Concept: Since we do not know $\mathscr{M}$ in advance, the abovementioned optimisation problem cannot be solved optimally. Instead, we use the following approach to approximate the optimum: As different tensor completion algorithms might return different tensors, even if we start with the same partially filled tensor, we use these differences to capture the uncertainty of our knowledge about the true tensor $\mathscr{M}$. In particular, in our approach, we choose 3 standard state-of-the-art tensor completion techniques to fill the tensor, and for each entry (i, j, k) within the tensor.[4] We define the cell value's uncertainty as the biggest difference between the 3 values these tensor completion methods provide for that particular entry. After this, for each elicitation question, we define its total uncertainty value by the sum of the uncertainties of the entries the question covers. We measure how much uncertainty can be reduced by asking that particular question). Finally, we use a knapsack model to identify the set of questions that maximises the uncertainty reduction, where each question is represented by an item within this knapsack model: The item's value is the question's total uncertainty, and the weight is the question's total annoyance cost, the annoyance cost budget $\mathscr{B}$ is the knapsack's capacity.[5]

Now, we use two heuristics to implement this knapsack based solution. In the *singleshot* heuristic, we solve this knapsack problem only once to identify all the questions at once. *Multishot*, on the other hand, sequentially asks one question per round, In particular, it starts with solving the knapsack problem as the singleshot version. But then instead of asking all the questions from the solution set of the knapsack, it only chooses the one with highest uncertainty reduction. After new entries are revealed from the answer of the chosen question, it uses the 3 tensor completion methods to reevaluate the new uncertainty values of the remaining empty entries in the tensor, and formulates a new knapsack in order to identify the next question to ask. It repeats this process until the annoyance budget is exceeded.

The advantage of the *multishot* heuristic, compared to its singleshot counterpart, is that it can be adaptive, and thus, more accurate. However, this improvement comes with an increased computational cost (as we have to solve multiple knapsack problems, each per round).

User Preference Changes: In our model, we also consider the case when the robot fails to successfully execute a certain task. This may occur due to the incorrect calculations for identifying the optimal physical motions for that task. If the human user observes that the robot cannot execute the task as demanded (e.g., even after many attempts), their preference for asking the robot to execute the same task in the future will be decreased. In this paper, we analyse the impact of this issue to the accuracy of the robot's preference estimation by investigating how the robot estimates user preferences in a multi-day task execution period, without having to run the full preference elicitation each day (for more details, see Section IV-B).

IV. Experimental Setup

To demonstrate the efficiency of our model, we test it in a realistic simulator, in which we simulate a typical office environment with a the TIAGo Steel robot and a number of human office users using Gazebo. In what follows, we first describe this simulated environment in more detail. We then discuss the parameter setup of our experiments.

A. Robot Simulator Environment

The simulated environment comprised two office rooms (3.5 m $\times$ 3 m each) and one common room (5 m $\times$ 3 m), with static office equipment (i.e., desks, chairs, food cupboard,

[3]Intuitively, α is intended to give a nearly linear bother curve for users with moderate willingness values (i.e., $w = 5$) while giving bother curves with more exponential and logarithmic appearances for more unwilling and willing users, respectively. The value of *Init* is intended to reflect the cost of bothering a user for the first time in which *Init* will be negligible (resp. quite high) for a very willing (resp. an unwilling) user.

[4]In this paper, we use tensor completion techniques from [20], [21], [22].

[5]For the description of knapsack problems, see, e.g., [23].

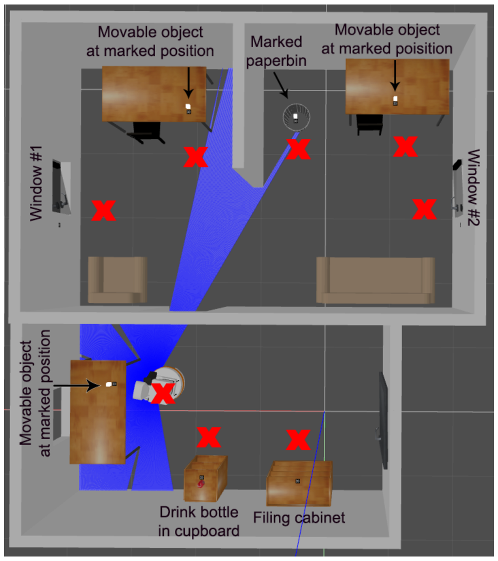


Fig. 1. The TIAGo robot in the office environment simulated with Gazebo, with points of interest for office tasks marked with red crosses. Blue shaded area indicate visibility for navigation of the robot's laser rangefinder.

filing cabinets, etc.), and dynamic objects (i.e., boxes, drink bottles and openable windows). Our environment provides several potential tasks that may be performed by TIAGo for the office workers, such as opening a window, lifting a box and fetching a drink. To perform these tasks, TIAGo has a pan-tilt head with a RGB-D camera, a lifting torso with a 7 DoF arm and a parallel gripper, all mounted on a differential drive mobile base equipped with laser rangefinder and ultrasound navigation sensors.

To autonomously navigate in its environment, TIAGo uses a map of the office containing the layout of rooms and the location of static objects (furniture). This map was created by navigating TIAGo around the office, while it mapped the area with its frontal laser rangefinder and 3 rear ultrasound sensors. Several points of interest – locations which are significant for performing tasks – were also saved in the map (e.g., a location on the floor next to the window from where the window may be opened, see Figure 1).

To perform office tasks, TIAGo navigates to the coordinates of an interest point, detects appropriately placed ArUco markers ([24]) with its RGB-D cameras, and uses them to track significant positions, for instance the location of a drink bottle or the handle of a window (see Figure 2). This enables TIAGo to move its 7 DoF arm to an object at a marked location, gripping that object with its parallel gripper end-effector, picking it up and moving it to different location. Using these functionalities, the following set of tasks were implemented, to be executed by TIAGo based on user (office worker) preferences:

- **T-1**: Bring a package to a user from the main desk.
- **T-2**: Collect an office document from one user and deliver to another.
- **T-3**: Fetch a drink from the cupboard.
- **T-4**: Open window #1 (right hinge).
- **T-5**: Open window #2 (left hinge).
- **T-6**: Collect files from user and file them.

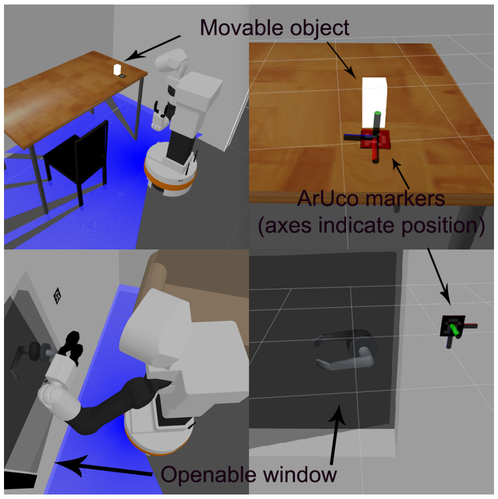


Fig. 2. Top: TIAGo locating an object to pick up; Bottom: TIAGo opening a window (Left: third-person point of view; Right: TIAGo's point of view).

- **T-7** Empty a paper bin.

The tasks commonly executed together in typical office situations were grouped into task clusters: break-time tasks (T-3, T-4, T-5), morning tasks (T-1, T-3, T-4 and T-5), end-of-day tasks (T-2, T-6 and T-7), and cleaning related tasks (T-4, T-5 and T-7). An office day was divided into time slots ($|T| = 8$ hours) and no more than one task was performed by TIAGo in each time slot.

B. Experimental Setup

Our study investigated the TIAGo robot assisting office workers in two types of experiments:

- **Single-day** experiments were performed over one day. Here, a full preference elicitation was executed at the beginning of the day (see details in section 3), following which TIAGo selected and executed a set of tasks for the day. On completing all its tasks at the end of the day, the users would update their preferences for each one of tasks, based on whether they had been completed successfully.
- **Multi-day** experiments were performed over five consecutive days, the first of which was identical to the single day experiment. From day two of the multi-day experiment, TIAGo would only perform a partial preference elicitation by asking a few questions at the beginning of the day, instead of executing a full preference elicitation (as done on the beginning of the first day). TIAGo would accordingly select and execute its tasks for that day. This was repeated until the end of the multi-day experiment.

Full Preference Elicitation: To perform a full preference elicitation, TIAGo selects an ideal set of questions Q^* (with Equation 3) using a chosen heuristic (Multishot, Oneshot or Random). It interacts with all the users, asks them the questions Q^* and elicits their preferences in M.

Changes in User Preference with Task Performance: At the end of each executed tasks, every user decides whether the task was performed successfully. In case of unsuccessful

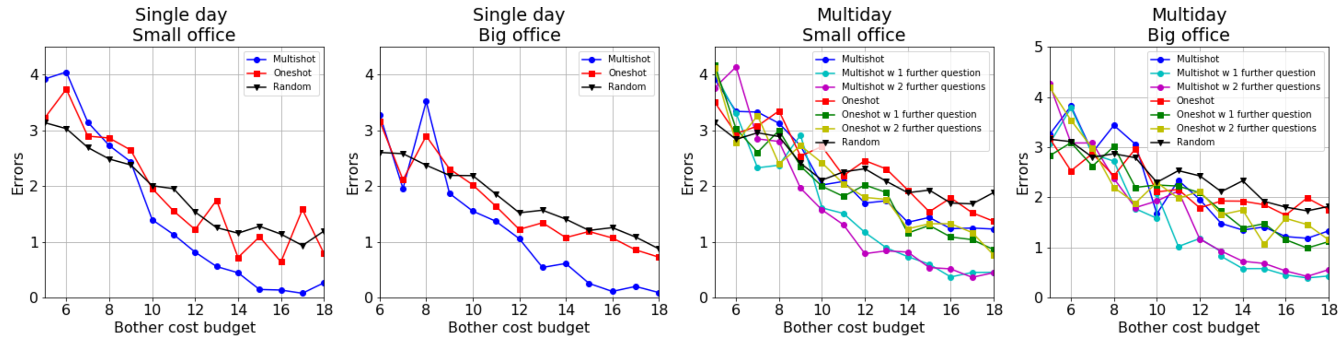


Fig. 3. Error results in small and big offices (single day experiments on the left, multi-day experiments on the right).

completion of a task, every user's preference value about the execution of that task decreases by d at that time slot. The preference values for the remaining time slots decrease by $d\gamma^{e(t)}$, where $\gamma \in (0,1]$ is the discount factor and $e(t)$ is the amount of time elapsed since completion of the unsuccessful task. In our experiments, γ was set to 0.8, and d to 2.

Partial Preference Elicitation: In the multi-day experiments the users also provide feedback to TIAGo on whether the task was completed successfully. At the beginning of day two, TIAGo asks a specified l number of questions about each unsuccessful task (performed the previous day) from a randomly selected user and updates its tensor M of learnt preferences and the optimal sequence of tasks to execute.

In both single and multi-day experiments, the set of tasks A was one of the task clusters selected at random. The questions for the user's preferences are inquired, on performing which of the $k \in \{1,\dots,|A|\}$ tasks in the selected cluster in which of $t \in \{1,\dots,|T|\}$ time slots. Overall, $|Q| = \lceil |A| * 11.5 \rceil$ questions were generated. Both the single and multi-day experiments were performed on small offices where the number of users $|U| \in \{5,\dots,10\}$ and in big offices where $|U| \in \{15,\dots,20\}$, were selected at random. The discount factor β was set to 0.95.

User Profiles for Experiments: For running our experiments, each office worker was randomly assigned one of the following four profiles (describing the way they provide their preferences):

- *random*: All preference values selected at random.
- *neutral*: All preference values are around the mid-values with no strong opinion expressed.
- *specific*: A specific time slot for a preference has a high value for a specific task, low preference values are provided otherwise.
- *normal*: Preference values have a normal distribution around the time slot most ideal for the user (the ideal time slot is selected at random).

V. Numerical Results

We first evaluated the proposed model for single day and multi-day scenarios considering the accuracy of eliciting user preferences. We then evaluated the performance of TIAGo in the simulation, examining the success of task executions.

Benchmarks: The preference elicitation of the single-day and multi-day model was evaluated on small ($\{5,\dots,10\}$) and big ($\{15,\dots,20\}$) offices, comparing the multishot and oneshot heuristics to a random one. The last one is a baseline benchmark which randomly picks a question and user to ask, until it exceeds the annoyance budget of that user. The evaluation considers several different annoyance cost budget sizes ($\{5,\dots,18\}$). The results show an error value averaged over 20 different randomly selected user configurations. The error value of a given solution is calculated based on the objective function of the optimisation problem (see Eq. (3)): $||W(Q,\mathscr{L})||_1/Z(Q))$.

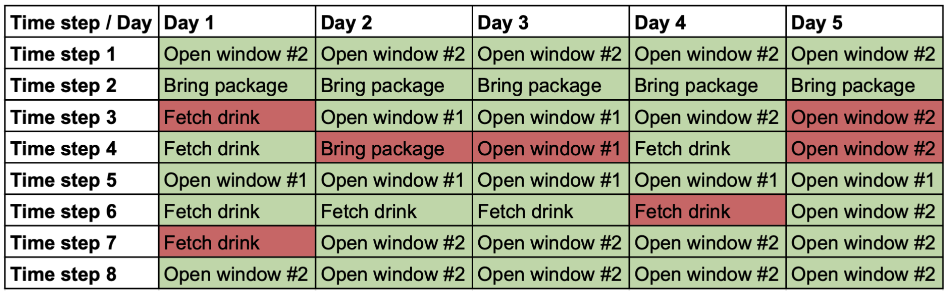
In case of multi-day models, the preference values elicited from the users were evaluated in comparison with the users' final day preference tensor. These models were evaluated for different number of additional questions asked: $l = [0,1,2]$.

Performance in Preference Elicitation: We performed single-day simulations using each one of the task clusters with different (randomly chosen) user combinations to evaluate the learning times, execution times of the tasks and the success rate of executing given tasks. Moreover, in case of multi-day simulations the sequence of optimal executed tasks and their success was logged; enabling us to see how this sequence changes based on user feedback over the week.

The benchmark results displayed in Figure 3 show the error against the annoyance cost budget ($\mathscr{B}$) for small and big office sizes in the single-day and multi-day scenario. It is clear that the solutions improve with the increase of the bother cost budget, as it allows TIAGo to ask more questions about the users' preferences. Considering the small office scenario, the multishot heuristic has the best performance, especially when the annoyance budget is larger than 9. When the budget is very tight, there is indeed not much room for improvement, compared to the baseline benchmark. When increasing the number of users (in the big office scenario), the errors do not change significantly compared to the small office case, showing that the algorithm scales well.

Number of Questions Asked: It is worth noting that in the single-day scenario, the multishot heuristic typically uses $3-6$ questions per user during the preference elicitation process, whereas singleshot and random typically requires $7-11$ (e.g., for $\mathscr{B} = 14$ multishot uses 6 questions, while singleshot and random use 11 and 8, respectively).

Multi-day Scenario: The multi-day models similarly show the decrease of error with the increase of the annoyance cost budget. Considering the order of algorithms in terms



Time step / Day	Day 1	Day 2	Day 3	Day 4	Day 5
Time step 1	🟩 Open window #2	🟩 Open window #2	🟩 Open window #2	🟩 Open window #2	🟩 Open window #2
Time step 2	🟩 Bring package	🟩 Bring package	🟩 Bring package	🟩 Bring package	🟩 Bring package
Time step 3	🟥 Fetch drink	🟩 Open window #1	🟩 Open window #1	🟩 Open window #2	🟥 Open window #2
Time step 4	🟩 Fetch drink	🟥 Bring package	🟥 Open window #1	🟩 Fetch drink	🟥 Open window #2
Time step 5	🟩 Open window #1	🟩 Open window #1	🟩 Open window #1	🟩 Open window #1	🟩 Open window #1
Time step 6	🟩 Fetch drink	🟩 Fetch drink	🟩 Fetch drink	🟥 Fetch drink	🟩 Open window #2
Time step 7	🟥 Fetch drink	🟩 Open window #2	🟩 Open window #2	🟩 Open window #2	🟩 Open window #2
Time step 8	🟩 Open window #2	🟩 Open window #2	🟩 Open window #2	🟩 Open window #2	🟩 Open window #2

Fig. 4. Executed tasks in a multi-day simulation (green: successful, red: unsuccessful). Unsuccessful task are preferred less often by the users.

of accuracy, the multishot, oneshot and random algorithms remain in the same order as in single-day cases, as they do not consider the following days changes. The multi-day multishot algorithm with 1 or 2 additional questions perform better with significant improvements. This, does not answer the question about what would be the optimal number of additional questions to ask, without spending a significant amount of annoyance cost in the subsequent days. Nevertheless, it is a proof of concept that further investigations are needed in this direction.

Performance of TIAGo in Office Tasks: Considering the performance of TIAGo in the simulations over 52 task executions, it has successfully completed 73% of the executed tasks. Task T-1 was the most successful on average, with 83% success rate, and T-7 the least successful at 33% success rate. The successfully executed tasks had a runtime of 148.05 seconds, compared to the unsuccessful tasks with the runtime of 173.78 seconds. This indicates that an unsuccessful task completion is often due to a subtask taking longer than expected and then failing. In most cases, the failed subtask fall into the following categories (in order of most to least probable source of error):

- After successfully lifting an object, TIAGo drops it while transporting it or before placing it down.
- TIAGo fails to pick up an object and topples it over instead.
- TIAGo does not navigate to the exact location of the task and thus cannot execute the task.

Figure 4 depicts an example on how the sequence of executed tasks change day by day, based on the success of their completion.

VI. Conclusions

We proposed a framework that combines tensor completion with knapsack algorithms to tackle the user annoyance issue in the process of preference elicitation for social robots. We also investigated the effect of incorrect/failure in task execution to the users' future preferences. Our findings show that while the user preferences can be accurately estimated in the single day scenario, there is still room to improve how to efficiently update the preference values without annoying users in the subsequent days.

References

[1] G. Canal, G. Alenyà, and C. Torras, "A taxonomy of preferences for physically assistive robots," in *IEEE RO-MAN*, 2017, pp. 292–297.

[2] D. Kragic, J. Gustafson, H. Karaoguz, P. Jensfelt, and R. Krug, "Interactive, collaborative robots: Challenges and opportunities." in *IJCAI*, 2018, pp. 18–25.

[3] A. Bestick, R. Pandya, R. Bajcsy, and A. D. Dragan, "Learning human ergonomic preferences for handovers," in *IEEE ICRA*, 2018, pp. 1–9.

[4] B. Woodworth, F. Ferrari, T. E. Zosa, and L. D. Riek, "Preference learning in assistive robotics: Observational repeated inverse reinforcement learning," in *Machine Learning for Healthcare*, 2018.

[5] N. C. Truong, T. Baarslag, S. D. Ramchurn, and L. Tran-Thanh, "Interactive scheduling of appliance usage in the home," in *IJCAI*, 2016.

[6] C. Rivoire and A. Lim, "The delicate balance of boring and annoying: Learning proactive timing in long-term human robot interaction," 2016.

[7] A. Garrell, M. Villamizar, F. Moreno-Noguer, and A. Sanfeliu, "Teaching robot's proactive behavior using human assistance," *International Journal of Social Robotics*, vol. 9, no. 2, pp. 231–249, 2017.

[8] L. M. Zintgraf, D. M. Roijers, S. Linders, C. M. Jonker, and A. Nowé, "Ordered preference elicitation strategies for supporting multi-objective decision making," in *AAMAS*, 2018, pp. 1477–1485.

[9] P. Dragone, S. Teso, and A. Passerini, "Constructive preference elicitation," *Frontiers in Robotics and AI*, vol. 4, p. 71, 2018.

[10] L. Sanneman, "Preference elicitation and explanation in iterative planning," in *IJCAI*, 2019, pp. 6456–6457.

[11] P. Kalgotra, R. Sharda, and R. McHaney, "Don't disturb me! understanding the impact of interruptions on knowledge work: an exploratory neuroimaging study," *Information Systems Frontiers*, vol. 21, no. 5, pp. 1019–1030, 2019.

[12] S. Rosenthal and M. Veloso, "Monte carlo preference elicitation for learning additive reward functions," in *IEEE RO-MAN*, 2012, pp. 886–891.

[13] T. Le, A. M. Tabakhi, L. Tran-Thanh, W. Yeoh, and T. C. Son, "Preference elicitation with interdependency and user bother cost," in *AAMAS*, 2018, pp. 1459–1467.

[14] T. Baarslag and M. Kaisers, "The value of information in automated negotiation: A decision model for eliciting user preferences," in *AAMAS*, 2017, pp. 391–400.

[15] T. Baarslag, A. T. Alan, R. C. Gomer, I. Liccardi, H. Marreiros, E. H. Gerding, and M. C. Schraefel, "Negotiation as an interaction mechanism for deciding app permissions," in *CHI*, 2016, pp. 2012–2019.

[16] N. C. Truong, J. McInerney, L. Tran-Thanh, E. Costanza, and S. D. Ramchurn, "Forecasting multi-appliance usage for smart home energy management," in *IJCAI*, 2013, pp. 2908–2914.

[17] M. W. Fleming, "Reasoning about interaction in mixed-initiative artificial intelligence systems," Ph.D. dissertation, 2004.

[18] Y. Ren, M. Qin, and W. Ren, "A web intelligent system based on measuring the effects of bother," in *WIC*, 2007, pp. 715–718.

[19] R. Cohen, M. Y. K. Cheng, and M. W. Fleming, "Why bother about bother: Is it worth it to ask the user," in *AAAI*, 2005.

[20] S. Gandy, B. Recht, and I. Yamada, "Tensor completion and low—rank tensor recovery via convex optimization," *Inverse Problems (27)*, 2011.

[21] B. Barak and A. Moitra, "Noisy tensor completion via the sum-of-squares hierarchy," in *COLT*, 2016, pp. 417–445.

[22] A. Montanari and N. Sun, "Spectral algorithms for tensor completion," *Communications on Pure and Applied Mathematics*, vol. 71, no. 11, pp. 2381–2425, 2018.

[23] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004.

[24] F. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and Vision Computing*, vol. 76, 2018.