Bayesian Active Learning for Collaborative Task Specification Using Equivalence Regions

Nils Wilde, *Student Member, IEEE,* Dana Kulić, *Member, IEEE,* and Stephen L. Smith, *Senior Member, IEEE*

***Abstract*—Specifying complex task behaviours while ensuring good robot performance may be difficult for untrained users. We study a framework for users to specify rules for acceptable behaviour in a shared environment such as industrial facilities. As non-expert users might have little intuition about how their specification impacts the robot's performance, we design a learning system that interacts with the user to find an optimal solution. Using active preference learning, we iteratively show alternative paths that the robot could take on an interface. From the user feedback ranking the alternatives, we learn about the weights that users place on each part of their specification. We extend the user model from our previous work to a discrete Bayesian learning model and introduce a greedy algorithm for proposing alternative that operates on the notion of equivalence regions of user weights. We prove that with this algorithm the revision active learning process converges on the user-optimal path. In simulations on realistic industrial environments, we demonstrate the convergence and robustness of our approach.**

I. Introduction

We address two active research topics in human-robot interaction (HRI): learning from non-expert users and human-robot collaboration. We develop a methodology for non-expert users to create specifications for complex robot tasks [1]. These specifications then enable humans and robots to operate in a shared workspace [2].

For instance, in an industrial environment shared between humans, autonomous and human-operated vehicles, a facility operator might define road rules to be followed by both robot and human-operated vehicles. Such road rules increase the predictability of robot behaviour for humans in the environment. These can include constraints such as areas of avoidance, one way roads or speed limits. The environment map, the operator specifications and a defined set of start and goal locations yield a complete specification of a robot task. In practice, designing such rules can be challenging, as operators might have little intuition about how their specification will affect the robot's behaviour and therefore the performance. They might be willing to accept the violation of less important constraints if sufficiently beneficial for task performance. For instance,

Manuscript received September, 10, 2018; Revised December, 8, 2018; Accepted January, 13, 2019.

This paper was recommended for publication by Editor Dongheui Lee upon evaluation of the Associate Editor and Reviewers' comments.

This research is partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and OTTO Motors.

N. Wilde and S. Smith are with the Department of Electrical and Computer Engineering, University of Waterloo, D. Kulić is with the University of Waterloo and Monash University. (nwilde@uwaterloo.ca; dana.kulic@uwaterloo.ca; stephen.smith@uwaterloo.ca)

Digital Object Identifier (DOI): see top of this page.

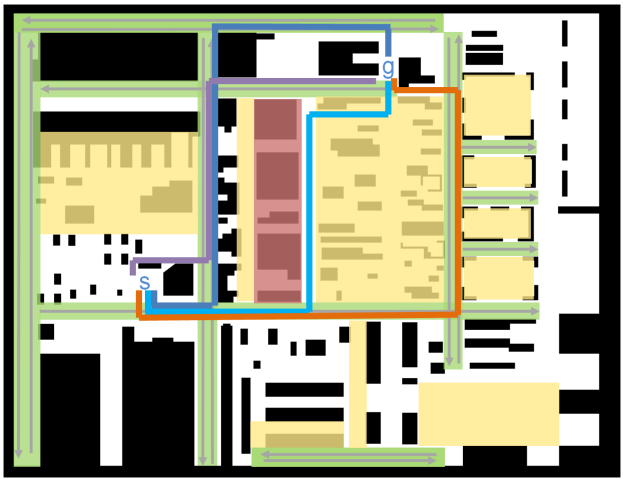


Fig. 1. Example environment (white) with obstacles (black) and user defined constraints. Roads are drawn in green with an arrow indicating the direction. Speed limit zones are drawn in yellow, while areas of avoidance are illustrated in red. Further, four different paths between a given start and goal are shown. Dark blue indicates the initial path following the specification. Purple and orange paths are alternatives that the simulated user accepted during the interaction. In cyan we show the user optimal path P^*, to which the learning eventually converged.

Figure 1 shows an industrial environment with several user defined constraints. When the robot uses the dark blue path, it respects all constraints. In the alternative solutions, the robot traverses (i.e., violates) constraints as this enables a significant reduction in the time to travel from start to goal.

A user likely has a preference for which path is a better solution for the given task, based on the completion time and on the importance of the constraints. This can be captured via weights, describing the importance of constraints. However, asking the user to define these weights is unintuitive and possibly challenging. We propose a framework where a user provides only a spatial definition of constraints, while the importance of each constraint is latent.

To fill the gap between this incomplete user input and a complete robot task description we apply active preference learning. We present the user with alternative solutions, i.e., paths, and ask for a ranking. In our framework, the user is "on-the-loop" and provides feedback at their convenience. The latest path preferred by the user becomes the current path and can already be executed. Consequently, each set of alternative paths contains the current path. Through this interaction with the user, the relative importance of each constraint can be learned. In our previous work [3], we developed an algorithm

that iteratively builds a set of linear inequalities on the hidden weights of the user constraints. Our user model evaluated paths based on a cost function trading-off constraint violation and time. The learning system assumed that the user would always provide feedback consistent with that cost function and thus iteratively rejected paths that became inconsistent with the user feedback. We extend our previous work [3] in two ways: First, the assumptions on the user are relaxed in order to capture more realistic behaviour. By considering noisy user feedback and introducing a probabilistic learning approach, we allow users to not always behave consistently with our user model. Second, based on our formulation of the problem as a shortest path search on a graph, we define equivalence regions for possible solutions. We propose a new learning system that exploits the notion of equivalence regions, which are sets of constraint weights that are indistinguishable to the user. From this we obtain a greedy algorithm that allows highly efficient learning, outperforming other state-of-the-art techniques.

Related work: Recently, active preference learning has been extensively studied for robot task specification. Thereby, a user is assumed to have an internal, hidden cost function which is learned from the user feedback to a presented set of alternatives. For instance, in [4] experts rank the demonstrated task performance for grasping applications while [5], [6] focus on continuous trajectories of dynamical systems like autonomous mobile robots. We propose a framework relying on a deterministic black-box planner that outputs a path for a given set of weights for the user constraints [3]. Different weights can have the same optimal solution, allowing for a discretization of the weight space. Therefore, our problem can be cast as an entity identification problem [7]: Our hypotheses are the sets of constraint weights that have different optimal solutions, tests correspond to asking the user about their preference between paths and observations equal their feedback. Golovin et al. [8] introduce a strong algorithm for near-optimal Bayesian active learning with noisy observations. However, their approach focuses on running each test at most once while we allow for repeated queries. While [5] greedily reduces the integral of the continuous probability density function over the weight space, our greedy algorithm is formulated over the discretized weight space corresponding to unique paths. A major drawback of the user model proposed in [5] and [6] is that the user's behaviour depends on the scale of the selected features as it considers an absolute instead of a relative error and does not provide a normalizing mechanism. We propose a more general, scale-invariant user model and show its robustness in simulations. Finally, our work differs from other applications of active preference learning for robotics in the way we choose the features for the cost function. Usually, features are picked manually and are therefore a design choice for the learning system [4], [9]. In our case, features are the violations of constraints that follow from the user specification and are user specific.

A common technique for learning from demonstration [10] is inverse reinforcement learning (IRL). The optimal behaviour of a dynamical system is described by a hidden reward function. IRL then learns this reward function by observing optimal demonstrations. Similarly, we assume a hidden user cost/reward function for the quality of a path. The cost function is modelled as a weighted sum of predefined features and we are interested in learning the weights. However, providing demonstrations might be difficult [4], the amount of necessary demonstrations may be prohibitively large [11] or demonstrations may require a high level of expertise [12]. Providing rich and precise specifications prior to a robot executing a task can be challenging and more prone to inaccuracies [13]. In contrast, active preference learning learns hidden reward functions by proposing alternative solutions and asking for the user's preference. Closely related to this work, [14] presents a GUI for specifying the user constraints on a given environment, which is used as a front-end to the work presented in this paper.

Contributions: In our previous work [3], we proposed a deterministic user model for learning about weights from a ranking feedback and proposed a complete algorithm. Using the same framework for combining path planning with user constraints, we extend the user model. To capture user feedback inconsistent with our assumed cost function, we propose a Bayesian learning approach (Section III-A). Thereby, we exploit the discrete properties of our problem, introducing a partitioning of the solution space based on equivalence regions. We prove almost sure convergence of the algorithm (Section III-B) and derive a greedy approach (Section III-C). Finally, we show the performance and robustness of our approach in comparison with another state-of-the-art technique in extensive simulations (Section IV).

II. Problem Formulation

A. Preliminaries

Using definitions from [15], a multi-graph is a triple $G = (V, E, \Psi)$, where the function $\Psi : E \to \{(v, w) \in V \times V : v \neq w\}$ associates each edge with an ordered pair of vertices. Multiple edges are allowed to connect the same ordered pair of vertices and are then called parallel. In our problem we consider doubly weighed multi-graphs of the form $G = (V, E, \Psi, c_1, c_2)$. Thereby, c_1 and c_2 are independent weight functions, each associating a real number to each edge of the graph: $c_i : E(G) \to \mathbb{R}$ for $i \in \{1, 2\}$.

A walk between two vertices v_1 and v_{k+1} on a graph G is a finite sequence of vertices and edges $v_1, e_1, v_2, e_2, \ldots, e_k, v_{k+1}$ where $e_1, e_2, \ldots e_k$ are distinct. A path $P_{v_1, v_{k+1}}$ between two vertices v_1 and v_{k+1} is defined as a graph $(\{v_1, v_2, \ldots, v_{k+1}\}, \{e_1, e_2, \ldots, e_k\})$ where $v_1, e_1, v_2, e_2, \ldots, e_k, v_{k+1}$ is a walk. On a weighted graph, the cost of a path is defined as $c(P) = \sum_{e \in P} c(e)$. In doubly weighted graphs we define two costs c_1 and c_2 where $c_i(P) = \sum_{e \in P} c_i(e)$ for $i \in \{1, 2\}$.

B. Problem statement

We summarize the problem setup in Figure 2. From the environment and a set of user constraints we can construct

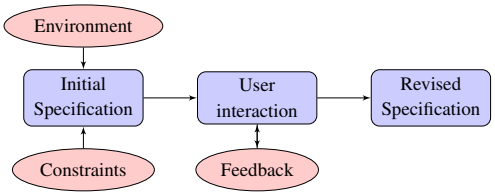


Fig. 2. Flowchart of the problem. An initial specification is revised using interactive learning to obtain a revision that better fits the user preferences.

an initial specification for the robot. As users might allow the violation of some of their constraints for sufficient time benefit, we present them with alternative solutions during the interaction and ask for feedback. From this feedback we learn the weights of the constraints and obtain a revised specification that corresponds to the user preferences. In Figure 1 we illustrate how the path that we believe to be optimal evolves after observing user feedback until it converges to the optimal solution.

As in our previous work [3], we consider a fully known, static environment, represented as a weighted strongly connected multigraph $G' = (V, E, \Psi, t)$. The weight t on the graph encodes the time a robot requires to traverse an edge. We use parallel edges with different times to model speed. A robot task consists of navigating from a start vertex v_{start} to a goal vertex v_{goal} on G'. On the environment, a user specifies a set Γ containing d constraints. Each constraint is a pair (E_i, w_i^*), where E_i is a subset of the edges of G' and w_i^* is a hidden user cost for the constraint. To incorporate the user specification, we create a doubly weighted graph $G = (V, E, \Psi, t, w^*)$. For each edge e in G the second weight $w^*(e)$ is defined as the sum of all w_i^* that belong to a constraint containing e. The problem is to find a path from v_{start} to v_{goal} that minimizes the following objective:

$$\min_{P} \sum_{e \in P} w^*(e) + t(e). \tag{1}$$

The true user weights w_i^* are latent. Moreover, they are defined in units of time, allowing us to pose the multi-objective optimization as an unweighted sum. To learn about the weights, we can query the user by presenting them with a set of paths $\{P^0, P^1, \dots P^k\}$. The feedback is a vector $\boldsymbol{u} \in \mathbb{R}^k$ representing a ranking, i.e., a partial ordering, of the presented paths. Without loss of generality we focus only on pair-wise comparisons, as the ranking of additional elements can be expressed with a set of pair-wise relations. This is also well motivated with respect to the user; ranking more than two alternatives might be unnecessarily challenging [16].

We formally define the *Learning of User Preferences* (LUP) problem as follows:

Problem 1 (LUP)**.** Given a graph G', a user specification Γ, a start and goal vertex, a user evaluating a presented set of paths and a budget of iterations for querying the user, maximize the belief about the true user weights w_i^* and its corresponding shortest path P^* with respect to equation (1).

III. Probabilistic Learning

In this section we propose a probabilistic model of user behaviour. In our previous work [3] we required the user to always provide feedback consistent with a linear user model. In contrast, we now consider that the user feedback may be noisy and thus the user feedback is not deterministic.

A. Bayesian Learning

Using definitions for Bayesian inference from [17] we set up a learning model for gaining information about the hidden (latent) parameter $\boldsymbol{w}$. We model the user weights to be positive and finite: $w_i \in [0, w^{\max}]$. The cost of a path P is $C(P) = \boldsymbol{\phi}\boldsymbol{w}^* + t$, where the violation vector $\boldsymbol{\phi}$ describes how many edges of each constraint are traversed by P, $\boldsymbol{w}^*$ is a column vector containing all latent user weights and t is the time to traverse P. From each user feedback for a pair (P^i, P^j) we can derive a hyperplane of the form $(\boldsymbol{\phi}^i - \boldsymbol{\phi}^j)\boldsymbol{w} = t^j - t^i$. This hyperplane defines two subsets of the weight space, Λ^{ij} and Λ^{ji}, where $\Lambda^{ij} = \{\boldsymbol{w} \in [0, w^{\max}]^d | (\boldsymbol{\phi}^i - \boldsymbol{\phi}^j)\boldsymbol{w} \leq t^j - t^i\}$. Thus Λ^{ij} is the set of all weights for which P^i has lower cost than P^j.

Probabilities of halfpspaces: For any pair of paths (P^i, P^j), the parameter $\boldsymbol{w}^* \in \Lambda^{ij}$ holds iff $C(P^i) \leq C(P^j)$. Adopting a Bayesian perspective, we treat $\boldsymbol{w}^* \in \Lambda^{ij}$ as a random variable and assign an uninformed prior $\mathbb{P}(\boldsymbol{w}^* \in \Lambda^{ij}) = 1/2$. Notice that the volumes of Λ^{ij} and Λ^{ji} do not correspond to the probability of a path being preferred over another path. From the user feedback about two paths P^i and P^j we obtain binary observations. We denote observations with a random variable U^{ij}, indicating whether the user prefers path P^i or P^j. A deterministic user always provides feedback where $U^{ij} = 1 \iff C(P^i) \leq C(P^j)$, i.e., $U^{ij} = 1 \iff \boldsymbol{w}^* \in \Lambda^{ij}$. A probabilistic user is consistent with this model with some probability p^{ij}. Hence, the probability of U^{ij} given $\boldsymbol{w}^* \in \Lambda^{ij}$ is

$$\begin{aligned} \mathbb{P}(U^{ij} = 1 | \boldsymbol{w}^* \in \Lambda^{ij}) &= \quad p^{ij} \\ \mathbb{P}(U^{ij} = 1 | \boldsymbol{w}^* \notin \Lambda^{ij}) &= 1 - p^{ij}. \end{aligned} \tag{2}$$

We refer to p^{ij} as the accuracy of the user and assume $p^{ij} > 1/2$, i.e., that our user model fits the user's decision making better than a random guess. If the parameter p^{ij} is hidden, we can evaluate equation (2) with an estimate $\hat{p}$. To simplify notation we write $\mathbb{P}(U^{ij} = 1)$ as $\mathbb{P}(U^{ij})$. In general, p^{ij} is a function of P^i and P^j. This allows us to model different levels of the user's accuracy depending on how similar the paths are.

Probabilities of equivalence regions: Equation (2) describes an observation model for a pair of paths, assigning probabilities to halfspaces. We now assign probabilities to paths instead. We observe that not every value in the weight space leads to a unique shortest path, which leads to our definition of equivalence regions.

Definition 1 (Equivalence region)**.** If the same path is optimal for two weights $\boldsymbol{w}^i$ and $\boldsymbol{w}^j$, we call $\boldsymbol{w}^i$ and $\boldsymbol{w}^j$ *equivalent*. An *equivalence region* of a weight $\boldsymbol{w}^i$ is then the set of all weights that are *equivalent* to the weight: $\Omega(\boldsymbol{w}^i) = \{\boldsymbol{w}^j \in \mathbb{R}^n_{\geq 0} | \boldsymbol{w}^j \text{ is } \textit{equivalent} \text{ to } \boldsymbol{w}^i\}$.

We can use equivalence regions to discretize the weight space $[0, w^{\max}]^d$. Given a comparison of two paths (P^i, P^j), we introduce a second observation model that describes the probability of user feedback given that the true user weight $\boldsymbol{w}^*$ lies in the equivalence region Ω' of some path P' as

$$\mathbb{P}(U^{ij}|\boldsymbol{w}^* \in \Omega') = \begin{cases} p^{ij}, & \text{if } \Omega' \subseteq \Lambda^{ij} \\ 1 - p^{ij}, & \text{if } \Omega' \subseteq \Lambda^{ji} \\ 1/2, & \text{otherwise.} \end{cases} \tag{3}$$

If an equivalence region lies in both halfspaces Λ^{ij} and Λ^{ji}, we obtain no information from the feedback U^{ij}, since not all weights in Ω' are either feasible or infeasible with the user feedback; expressed in the third case. Let $\mathbb{O}$ be the set of all equivalence regions for a given problem instance. The observation model allows us to express a probability for $\boldsymbol{w}^* \in \Omega'$ given an observation U^{ij} as a Bayesian posterior

$$\mathbb{P}(\boldsymbol{w}^* \in \Omega'|U^{ij}) = \frac{\mathbb{P}(U^{ij}|\boldsymbol{w}^* \in \Omega')\mathbb{P}(\boldsymbol{w}^* \in \Omega')}{\sum\limits_{\Omega \in \mathbb{O}} \mathbb{P}(U^{ij}|\boldsymbol{w}^* \in \Omega)\mathbb{P}(\boldsymbol{w}^* \in \Omega)}. \tag{4}$$

Following [17], we write the Bayesian posterior for a series of n observations U for arbitrary pairs of paths as

$$\mathbb{P}(\boldsymbol{w}^* \in \Omega'|U) = \frac{\prod\limits_{U^{ij} \in U} \mathbb{P}(U^{ij}|\boldsymbol{w}^* \in \Omega')\mathbb{P}(\boldsymbol{w}^* \in \Omega')}{\sum\limits_{\Omega \in \mathbb{O}} \prod\limits_{U^{ij} \in U} \mathbb{P}(U^{ij}|\boldsymbol{w}^* \in \Omega)\mathbb{P}(\boldsymbol{w}^* \in \Omega)}. \tag{5}$$

Remark. Notice that this general model does not depend on the exact form of the likelihoods p^{ij}, we only require $p^{ij} > 1/2$. Therefore, our model could use the likelihood function from [6]. Alternatively, one could fix all p^{ij} to a constant. Then, in contrast to [5], [6], the accuracy of the user does not depend on the scaling of the features in the cost function. Moreover, our model increases the robustness towards user feedback that appears inaccurate because the user is considering context that is not described by our features. For instance, in a warehouse an operator might have different preferences for different weekdays or wants a robot to temporarily avoid certain regions. This can not be covered with the current cost function and thus this user would appear erratic to the learner. Finally, when the accuracy is set to one, the deterministic learning model [3] is recovered. The key advantage of using equivalence regions in equation (5) is that it reduces the complexity of the probability distribution since we now have a discrete distribution over regions rather than a continuous one This allows for a significantly faster solving of the problem, as we will show in Section IV.

B. Probabilistic Algorithm

In Algorithm 1 we propose a general procedure to iteratively learn about user preferences from pairwise user feedback with inaccurate users. Initially, we compute the set of all equivalence regions $\mathbb{O}$ (line 2). After updating our current belief about all equivalence regions (line 4), we iteratively generate new paths (line 5) similar to our deterministic algorithm from [3]. Then, we request user feedback for the pair $(P^{\text{curr}}, P^{\text{new}})$ and add the user feedback to a set (6-7). After adding the new observation to our set, we update the weight space and, if necessary, the current weight (8-9). The procedure is repeated until we reach the iteration budget N in line 2, at the end we return the weight $\hat{\boldsymbol{w}}^{\text{best}}$ where the posterior belief is maximized. We discuss an implementation of the function `getNewPath`$(\cdot)$ in Section III-C.

Algorithm 1: Learning user weights by sampling

Input: G', Γ, N
Output: $\hat{\boldsymbol{w}}^{\text{curr}}$

1 $\hat{\boldsymbol{w}}^{\text{curr}} = \boldsymbol{w}^{\max}$, $U = \emptyset$
2 Calculate $\mathbb{O}$
3 **for** $n = 1$ *to* N **do**
4 Update $\mathbb{P}(\boldsymbol{w}^* \in \Omega'|U)$ for all $\Omega' \in \mathbb{O}$
5 $P(\hat{\boldsymbol{w}}^{\text{new}}) \leftarrow$ `getNewPath`$(\mathbb{O}, \hat{\boldsymbol{w}}^{\text{curr}}, \{\mathbb{P}(\boldsymbol{w}^* \in \Omega^1|U), \mathbb{P}(\boldsymbol{w}^* \in \Omega^2|U), \dots\})$
6 Get user feedback $U^{\text{curr,new}}$ for paths $P(\hat{\boldsymbol{w}}^{\text{curr}})$ and $P(\hat{\boldsymbol{w}}^{\text{new}})$
7 $U = U \cup U^{\text{curr,new}}$
8 **if** $U^{\text{curr,new}} = \text{new}$ **then**
9 $\hat{\boldsymbol{w}}^{\text{curr}} = \hat{\boldsymbol{w}}^{\text{new}}$
10 **return** $\hat{\boldsymbol{w}}^{\text{best}} = \underset{\boldsymbol{w}'|\Omega' \in \mathbb{O}}{\arg\max}\, \mathbb{P}(\boldsymbol{w}^* \in \Omega'|U)$

Convergence: We now establish almost surely convergence of Algorithm 1. Let $\boldsymbol{w}^*$ be the true user weight and $p^{ij} > 0.5$ for all pairs of paths (P^i, P^j). Without loss of generality, we only consider i, j pairs that are ordered such that $C(P^i) \leq C(P^j)$; hence $\boldsymbol{w}^* \in \Lambda^{ij}$ always hold (but this is not known to the algorithm). Moreover, l is the number of equivalence regions in $\mathbb{O}$ and m the number of all pairwise comparisons. For the following definition we change our notation and denote the optimal path P^* as P^1.

Definition 2 (Asymptotically completely informative sequence)**.** Let X_n be a sequence of pairs of paths presented to the user in n iterations, and for each j, let $X^{1j} = \{X_1^{1r_1}, X_2^{1r_2}, X_{n_j}^{1r_{n_j}}\}$ be the longest subsequence of X for which $\Omega^j \subseteq \Lambda^{1r_1}, \Omega^j \subseteq \Lambda^{1r_2}, ..., \Omega^j \subseteq \Lambda^{1r_{n_j}}$. Then the sequence is asymptotically completely informative if as n goes to ∞, we have $n_j \to \infty$ for all j.

In other words, if a sequence of pairs of paths contains observations about every (P^*, P^j), and the number of observations for each (P^*, P^j) pair goes to infinity as the length of the sequence goes to infinity, it is called *asymptotically completely informative*. Notice that such a sequence of paths is not required to contain subsequences for all pairs of paths (P^*, P^j); it is sufficient if the feedback to the pairs (P^*, P^j) contains information about all other equivalence regions according to (3). Finally, we call U *asymptotically completely informative* if the corresponding sequence of paths is *asymptotically completely informative* and treat the probability that the true user weight $\boldsymbol{w}^*$ or an estimate $\hat{\boldsymbol{w}}$ lie in an equivalence region Ω as a random variable.

Proposition 1 (Convergence)**.** Let Ω^* be the equivalence region containing $\boldsymbol{w}^*$. Given *asymptotically completely informative* user feedback U, the probability that the best estimate $\hat{\boldsymbol{w}}^{\text{best}}$ of Algorithm 1 lies in Ω^* converges almost surely to 1 as all n_j go to infinity:

$$\mathbb{P}(\hat{\boldsymbol{w}}^{\text{best}} \in \Omega^* | U) = 1,\ n_j \to \infty, \text{ for all } j. \tag{6}$$

Proof. At first consider the comparison of an arbitrary pair (P^i, P^j) and fix $P^i = P^*$. Let U be a sequence of user feedback of length $n^{ij}$and k^{ij} be the number of times the user chooses P^i, i.e., chooses accurately. Moreover, let $\hat{p}^{ij}$ be our estimate of p^{ij}. For simplification we drop the ij superscript. From $p > 1/2$, we can conclude that $k > n/2$ as $n \to \infty$, using Hoeffding's inequality [18]. We notice that the probability for a sequence of user feedback given $\boldsymbol{w}^* \in \Lambda^{ij}$ depends on p, while our belief about $\boldsymbol{w}^* \in \Lambda^{ij}$ given some user feedback is based on $\hat{p}$. Given user feedback U with known n and k, the posterior probability is

$$\begin{aligned}\mathbb{P}(\boldsymbol{w}^* \in \Lambda^{ij} | U) =& \frac{\hat{p}^k (1-\hat{p})^{n-k}}{\hat{p}^k (1-\hat{p})^{n-k} + \hat{p}^{n-k}(1-\hat{p})^k} \\ =& \frac{1}{1 + \frac{\hat{p}}{1-\hat{p}}^{n-2k}}.\end{aligned} \tag{7}$$

We take the limit as n goes to ∞:

$$\lim_{n\to\infty} \mathbb{P}(\boldsymbol{w}^* \in \Lambda^{ij} | U) = 1 = \lim_{n\to\infty} \frac{1}{1 + \frac{\hat{p}}{1-\hat{p}}^{n-2k}}. \tag{8}$$

Using $\hat{p} > 1/2$ leads to $\frac{\hat{p}}{1-\hat{p}} > 1$. The term $n - 2k$ is strictly negative if $k > n/2$. Hence, $\hat{p}/(1-\hat{p})^{n-2k}$ approaches zero as n goes to infinity. We conclude that $\lim_{n\to\infty} \mathbb{P}(\boldsymbol{w}^* \in \Lambda^{ij} | U) = 1$. As we only have two paths, we only have two equivalence regions. Following our ordering of i and j, $\Omega^* = \Lambda^{ij}$. Hence, $\mathbb{P}(\hat{\boldsymbol{w}} \in \Omega^* | U) = 1$, as $n^{ij} \to \infty$ for a single, fixed pair of paths P^i and P^j.

Finally, we extend the result to comparisons for multiple pairs. Equation (5) expresses the probability of $\boldsymbol{w}^*$ lying in a given equivalence region. Notice that $\bigcap_{j\neq i} \Lambda^{ij} \subseteq \Omega^*$, as well as $\lim_{n\to\infty} \mathbb{P}(\boldsymbol{w}^* \in \Lambda^{ij} | U) = 1$. As $\mathbb{P}(\boldsymbol{w}^{\text{best}} \in \Lambda^{ij} | U) \to 1$ for all $j \neq i$, we have $\mathbb{P}(\boldsymbol{w}^{\text{best}} \in \bigcap_{j\neq i} \Lambda^{ij} | U) \to 1$. Hence, $\mathbb{P}(\hat{\boldsymbol{w}}^{\text{best}} \in \Omega^* | U) \to 1$ and the statement holds. □

From Proposition 1 we conclude that Algorithm 1 always elicits the true user weight if an *asymptotically completely informative* sequence of paths is presented to the user for feedback, and if the user's accuracy with respect to our model is greater than $1/2$. However this does not include any guarantees on the speed of convergence. In the next section we derive a greedy approach to maximize convergence speed.

C. Greedy Policy

We now show how to find new paths in each iteration of Algorithm 1, i.e., the function `getNewPath`. We notice that computing the set of all equivalence regions for a given problem instance is computationally intractable and the set can be of exponential size in relation to the number of constraints (see Proposition (1) in the supplementary material). Because of this, we propose a greedy algorithm for finding new paths. We define $q(\boldsymbol{w}^* \in \Omega | U)$ as the unnormalized posterior, i.e., the numerator of equation (5). As q is not a probability we refer to it as the *posterior measure*. The decrease in the posterior measure is captured as

$$f(X_n) = 1 - \sum_{\Omega^i \in \mathbb{O}} q(\boldsymbol{w}^* \in \Omega^i | U_n). \tag{9}$$

Our primary motivating application is one in which the user is "on-the-loop". We do not require them to constantly provide feedback and already execute the current solution P^{curr} [3]. Therefore, we keep the current best path and fix P^{curr} to be one of the two alternative paths comprising the next query (Algorithm 1). Thus, our greedy algorithm returns the path maximizing the posterior measure

$$P_n^{\text{new}} = \arg\max_{P^j} \quad \mathbb{E}\left[f\left((P^{\text{curr}}, P^j) \bigcup X_{n-1} \right) \right]. \tag{10}$$

In this optimization we only need to consider one P^j for each equivalence region. In the supplementary material we discuss the case where two new paths are presented, i.e., we do not fix one path in each query to be P^{curr}. In this case it can be shown that (10) is an adaptive submodular function. Finally, we ensure convergence for the greedy approach.

Lemma 1 (Convergence of the greedy algorithm)**.** The greedy algorithm equation (10) returns an *asymptotically completely informative* sequence of paths if the number of iterations goes to infinity and thus the probability of the true user weight converges to one almost surely.

Proof. To prove the statement we show two properties: 1) The greedy algorithm eventually returns P^* and 2) if $P^{\text{curr}} = P^*$ it eventually returns all paths necessary to constitute an *asymptotically completely informative* sequence. To show the first statement let $P^{\text{curr}} \neq P^*$. If a path $P^j \neq P^*$ is returned we either do not learn about P^* and the posterior of either Ω^{curr} or Ω^j decreases relatively to the posterior of Ω^* (see equation (3)). Otherwise, the comparison of (P^{curr}, P^j) contains information about Ω^* and thus is expected to increase the posterior of Ω^*. While $P^{\text{curr}} \neq P^*$, the expected marginal reward of presenting P^*, i.e., $\mathbb{E}\left[f\left((P^{\text{curr}}, P^*) \bigcup X_{n-1}\right)\right] - f(X_{n-1})$, increases monotonically, relatively to the reward of any other path. Thus, P^* will eventually be the maximizer of equation (10). Then, the greedy algorithm returns P^* and the user will prefer P^* over the current P^{curr} in expectation.

For the second statement, assume we already have $P^{\text{curr}} = P^*$. Due to inaccurate user feedback another path P^i becomes P^{curr}. However, as shown above, the algorithm eventually presents P^* again. Consider the path P^j where n_j is minimal among all paths (n_j is defined in Definition 2). Case 1: $q(\boldsymbol{w}^* \in \Omega^j | U)$ is the maximizer of equation (10), thus P^j is presented and n_j increments. Case 2: Some $q(\boldsymbol{w}^* \in \Omega^r | U)$ is the maximizer and (P^{curr}, P^r) is presented. If the corresponding feedback contains information about P^j, n_j increments as well. According to equation (3), if no information about P^j is obtained, $q(\boldsymbol{w}^* \in \Omega^j | U)$ increases by a ratio of $0.5/(1-p)$ (where p is the user accuracy) relative to all $q(\boldsymbol{w}^* \in \Omega' | U)$, where Ω' gets rejected by the feedback to (P^{curr}, P^r). As

this holds for all other paths and the number of paths is finite, $q(\boldsymbol{w}^* \in \Omega^j|U)$ increases relative to all other posterior measures until, after a finite number of iterations, either information about P^j is obtained and n_j increments, or case 1 applies. Hence, we are guaranteed to increment the minimal n_j and thus all n_j must go to infinity. □

Performing an exact greedy step is hard, as finding the set of all equivalence regions $\mathbb{O}$ is intractable. In practice, a polynomial sized estimate $\mathbb{O}'$ can be found via sampling which allows for an approximate greedy step.

IV. Evaluation

To generate realistic simulations, we recruited users to create specifications. Given the layout of a real industrial facility, users defined constraints as described in [14]. To systematically evaluate our approach, we simulate user feedback in the active learning. This allows us to pick different ground truths $\boldsymbol{w}^*$ and generate the user feedback with varying accuracy levels. Figure 1 illustrates an example specification with different possible solutions. Further, for an outdoor scenario we conducted experiments using graphs generated by a probabilistic method [19] and random specifications.

Our primary interest is how the posterior belief about $\boldsymbol{w}^*$ evolves. In the evaluation we have two objectives: Showing the robustness of our user model and comparing our work with [5]. We refer to their approach as the *Maximum Volume Removal (MRV)* and name our approach the *Maximum Equivalence Region Removal (MERR)* [1].

In our implementation of MRV, we modify the query selection: As we consider a discrete space, queries are found by iterating over $\mathbb{O}'$ rather then solving a continuous optimization problem. To ensure comparability with our framework, we fix one path in the pair that comprises a query as the current path. Moreover, MRV requires a scaling of the features. The user's accuracy is modelled as an exponential function of the difference in the cost of two paths [5]. Thus the user's accuracy depends on the scaling of the cost function. The model is extended by a linear parameter β in [6] to describe different levels of accuracy. However, as no restriction on the scale of the features is made, these β values do not yield similar results for different scenarios. In our experiments we manually determined β for each scenario such that the user's accuracy is approximately 0.9.

To investigate the performance we used three different specifications that vary in complexity. The first specification consists of 26 constraints, covering 33% of the free space, the second (shown in Figure 1) has 41 constraints covering 40% while the third consists of 52 constraints covering 73%[2]. For each specification we varied between two start and goal pairs and three randomly selected true user weights $\boldsymbol{w}^*$. As finding the set of all equivalence regions $\mathbb{O}$ is intractable, we generate estimates $\mathbb{O}'$ via sampling. The graph G' for these

[1]Note: In both approaches neither volume nor equivalence regions are removed, we rather assign a lower posterior probability to the rejected items.

[2]When a user specifies a road on the interface it counts as 2 constraints for the planner: A reward for following the road, i.e., a constraint with a negative weight, and a penalty for going against the direction of travel.

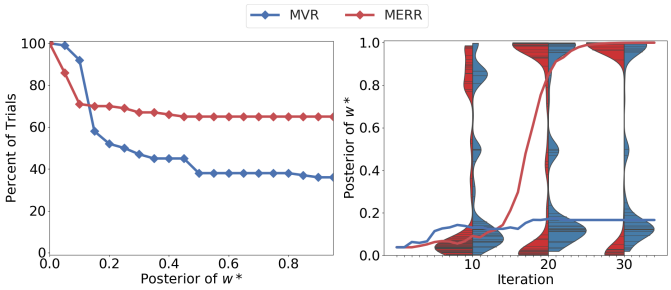


Fig. 3. Results for experiment 1. For different values of the posterior of the optimal weight $\boldsymbol{w}^*$ the left plot shows the percentage of trials that achieved that value within 30 iterations. In the right plot we show median values of the posterior over all 30 iterations together with violin plots of the distribution of the posterior of $\boldsymbol{w}^*$ at iterations 10, 20 and 30.

experiments is based on a grid layout and has of 5185 vertices. MERR and MVR differ in three components: The model for the simulated user feedback, the user model assumed by the learning system and the strategy for presenting new paths, i.e., the query selection.

A. Performance of MERR and MVR query selections

Experiment 1: In the first experiment, we compare the performance for the different active query selections - either MVR or MERR - for a user following the MVR model. Figure 3 shows the result for a total of 180 trials (10 repetitions for each configuration of user, start-goal pair and true user weight) with a budget of 30 iterations.

The left plot of Figure 3 illustrates the percentage of trials that achieved a given final posterior value for the true user weight w^* within the iteration budget. A critical threshold for the posterior is 0.5, as then $\hat{\boldsymbol{w}}^{\text{best}} = \boldsymbol{w}^*$ and $\hat{\boldsymbol{w}}^{\text{best}}$ is the unique maximizer of the posterior distribution. The MERR query selection has a higher success rate: 70% of the trials converge within 30 iterations, while only 40% do so for MVR queries. Interestingly, both approaches always converge once the posterior of $\boldsymbol{w}^*$ surpasses 0.5. In the right sub plot we illustrate the evolution of the median posterior of $\boldsymbol{w}^*$ over the iterations. Further, at iterations 10, 20 and 30 we show the distribution of the data. We observe that between iterations 10 and 20 the MERR posterior median starts to increase quickly, passing the 0.5 threshold at iteration 17 and getting past 0.9 after 21 iterations. MVR shows a slower increase and does not pass 0.2 within the 30 iterations. The violin plots at three stages of the process illustrate a further detail: The distributions are nearly bimodal. Both approaches succeed for some instances very quickly while the posterior stays low for harder instances. However, we observe that the low end of the distribution shrinks more quickly for MERR and eventually becomes completely bimodal. In some hard instances, the algorithm takes longer to initially show P^* to the user and does not learn about $\boldsymbol{w}^*$ until then. However, once P^* is shown, the belief about $\boldsymbol{w}^*$ is maximized quickly, leading to the bimodal distribution.

To explain the better performance of MERR, we recall that equivalence regions vary drastically in volume. MVR often proposes queries that reduce the integral of the posterior but

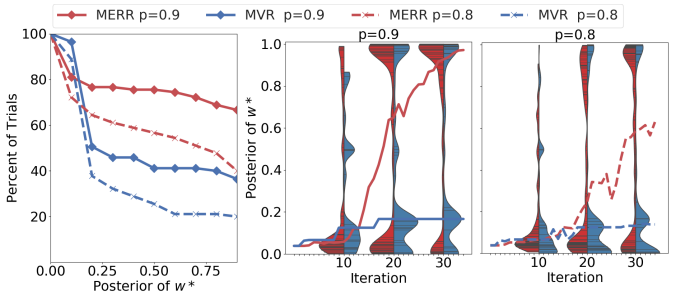


Fig. 4. Results for experiment 2. The same analysis as in Figure 3 is shown, but for the MERR user two different values for accuracy, $p = 0.9$ and $p = 0.8$, are depicted.

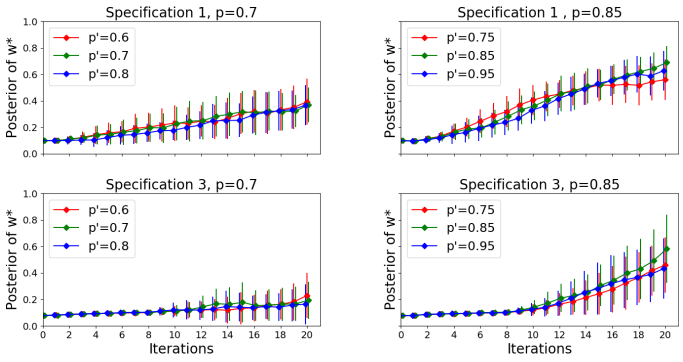


Fig. 5. Experiment 3. Robustness of the MERR user model for two specification (26 and 52 constraints) and different accuracies p in the simulated user and different estimates p' for the learning.

do not significantly change the posterior of equivalence regions and thus makes little progress.

Experiment 2: The second experiment focuses on the performance of both query selection methods, assuming the user generates responses according to the MERR model, i.e., equation (3). In Experiment 1 we simulated the user according to [5], [6]. In that model, the accuracy depends on how different the presented paths are and therefore is influenced by the query selection. In order to ensure comparability with our user model, we fixed the accuracy to the average of Experiment 1, thus $p = 0.9$. Additionally, a second dataset shows the results for lower accuracies of $p = 0.8$. Notice that the range for meaningful values is $(0.5, 1]$; users with $p = 0.5$ act completely independently of our model. Further, $p = 0.9$ corresponds to 10% misleading feedback, $p = 0.8$ doubles the error rate to 20% of cases. In Figure 4 we summarize the data as done for Experiment 1.

In contrast to Experiment 1, both query selection methods achieve a lower convergence rate for $p = 0.9$. MERR reaches a posterior median of 0.6 in 80% of the trials, while with MVR less than 45% of trials reach the 0.5 threshold. For the lower accuracy of $p = 0.8$, both query selection methods perform worse: MERR converges only in 42% and MVR 20% of cases. The graphs illustrating the mean posteriors over the iterations show a similar result for $p = 0.9$ compared to Experiment 1. For $p = 0.8$, the median of the posterior for MERR makes substantially better progress than MVR. The distributions confirm this observation: MERR shrinks the bottom lobe and gains on the upper end more quickly.

In summary, the first two experiments highlight the performance benefit when maximizing the decrease in the posterior summed over equivalence regions compared to the decrease in the integral of the posterior (i.e., the removed volume), irrespective of the user model.

B. Robustness of MERR

We further investigate how sensitive the proposed approach is to knowledge of the user's accuracy. We simulate the user according to the MERR model with a constant accuracy p, but the learning system only has access to an estimate $\hat{p}$. We fixed $p = 0.7$ and $p = 0.85$ and picked either $\hat{p} = p$, or $\hat{p} = p \pm 0.1$. The experiment is based on the smallest and largest specifications with 26 and 52 constraints, respectively. For each $p, \hat{p}$ configuration we average over 20 trials.

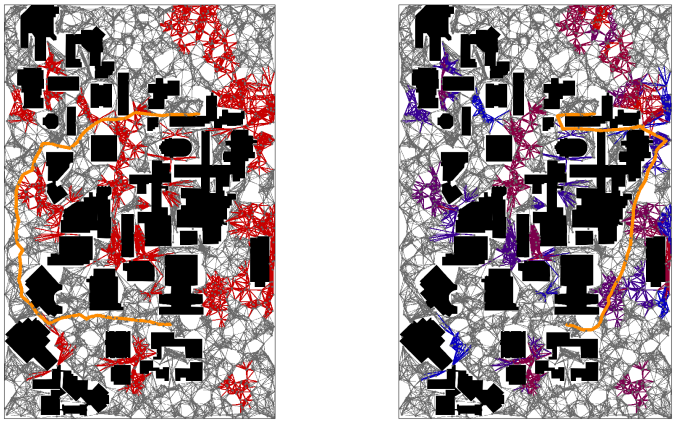
Figure 5 shows that an estimation error of 0.1 for the user accuracy has very little influence on the performance. In all 4 plots the over- and underestimates behave similarly to when the learner knows the user's accuracy exactly. As expected, the accuracy itself has an impact on the performance: For $p = 0.7$ the learning system performs worse for both specifications. Especially for the large specification, there is only little progress over the 20 iterations. In a more complicated setting additional feedback is needed to elicit the true user weight, the higher amount of inaccurate feedback then has a larger impact. On the other hand, for $p = 0.85$ the final result is relatively similar for both specifications. We conclude that a richer specification has a smaller impact on the performance when the user feedback is more accurate.

C. Extension to other scenarios

Finally, we applied the approach to a different setting: An environment described by a graph based on a k-nearest probabilistic roadmap (PRM) [19]. User specifications are generated randomly by sampling polygons in the environment; for each region a constraint is formulated over all edges incident with a vertex in the region. Using the layout of the campus of the University of Waterloo, we generated a PRM graph with 2000 vertices and $k = 10$, the number of sampled constraints is 50. In Figure 6 we show the map together with the generated graph. Further, we compare the initial path of the learning which does not violate any constraints, and the user optimal path P^* that is learned through interaction. We conducted a similar analysis as in Experiments 1 and 2, averaged for 20 different specifications. Overall 50% of the trials achieve a posterior of at least 0.9 within 50 iterations. Moreover, after 15 iterations the median passes the 0.5 threshold, while 0.9 is reached after 40 interactions.

V. Discussion

User models: Generally, both models, MVR and MERR, assume that the user evaluates paths based on a weighted sum of features. In MVR the user's accuracy depends on how similar the presented paths are. This approach is well motivated and promises good performance when the features are adequate. On the other hand, it has disadvantages with the



(a) (b)

Fig. 6. Results for Experiment 4. (a) shows the outdoor environment (buildings in black and freespace in white) with the generated PRM-graph. Red indicates edges that belong to a randomly generated constraint, orange shows the optimal path between a start and goal location of a task when not violating any constraints. (b) shows the optimal path P^* and the updated weights on the constraints – red indicates high and blue low weights.

scaling of features and lacks robustness when users do not follow the model. In contrast, our approach generally models inaccuracies as a random noise and is agnostic towards their exact form. In the simulations we fixed p to a constant and demonstrated the robustness. Therefore, our learning system is less dependent on our user model exactly capturing the real user behaviour. A limitation of the MERR user model with constant accuracy values is that accurate user feedback is potentially not exploited efficiently as the noise then is query-independent. This approach is investigated in [9]. Moreover, both learning models depend on sampling to perform the greedy step. Even though the accuracy can be increased arbitrarily with more samples, finding the optimal solution is computationally intractable.

Performance in experiments: In the first two experiments we showed data that was collected for different user weights and different user specifications (and thus features). Both have a direct influence on the algorithm's performance. Usually, more complex specifications have a larger set of equivalence regions, which affects the convergence. Maybe surprisingly, the true user preference $\boldsymbol{w}^*$ can also influence the performance, especially in the MERR model: The learning system makes little progress if most of the hyperplanes learned from a sequence of user feedback intersect Ω^*. Moreover, in single trials both models can perform relatively poorly by random chance. An inaccurate user feedback for a query containing P^* leads to decrease in the posterior of $\boldsymbol{w}^*$. Then, the learning system might need multiple iterations to present P^* again and thus elicit $\boldsymbol{w}^*$. This effect is enhanced when the accuracy of the user is low. Nonetheless, the MERR learning model is still guaranteed to converge.

VI. Conclusions and Future Work

We presented an interactive framework for robot task specification. Based on Bayesian active leaning we derived a greedy algorithm for generating queries. Our approach exploits the fact that different weights for constraints do not necessarily lead to different optimal paths. Using equivalence regions allows for a discrete Bayesian learning model that does not require the user to always provide feedback consistent with the assumed cost function. The probability of an inconsistent user feedback is of a general form and scale-invariant. In simulations, we demonstrated that our approach outperforms a related state-of-the-art technique [5] and showed robustness of our user model. One future work direction is to extend the concept of equivalence regions to continuous spaces by introducing a notion of path similarity. Further, the proposed framework can be applied to more complex and realistic scenarios including multiple start and goal locations and additional features, potentially including dynamic data such as traffic. Finally, user studies are required to show the practical performance of the framework.

References

[1] V. Villani, F. Pini, F. Leali, and C. Secchi, "Survey on human-robot collaboration in industrial settings: Safety, intuitive interfaces and applications," *Mechatronics*, no. June 2017, pp. 1–19, 2018.

[2] M. C. Gombolay, R. J. Wilcox, and J. A. Shah, "Fast scheduling of robot teams performing tasks with temporospatial constraints," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 220–239, 2018.

[3] N. Wilde, D. Kulic, and S. L. Smith, "Learning user preferences in robot motion planning through interaction," in *ICRA*, 2018.

[4] C. Daniel, M. Viering, J. Metz, O. Kroemer, and J. Peters, "Active Reward Learning," *RSS*, vol. 10, no. July, 2014.

[5] D. Sadigh, A. D. Dragan, S. Sastry, and S. A. Seshia, "Active preference-based learning of reward functions," in *RSS*, 2017.

[6] C. Basu, M. Singhal, and A. D. Dragan, "Learning from richer human guidance: Augmenting comparison-based learning with feature queries," in *HRI 2018*. ACM, 2018, pp. 132–140.

[7] V. T. Chakaravarthy, V. Pandit, S. Roy, P. Awasthi, and M. Mohania, "Decision trees for entity identification: Approximation algorithms and hardness results," in *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 2007, pp. 53–62.

[8] D. Golovin, A. Krause, and D. Ray, "Near-optimal bayesian active learning with noisy observations," in *NIPS*, 2010, pp. 766–774.

[9] R. Holladay, S. Javdani, A. Dragan, and S. Srinivasa, "Active comparison based learning incorporating user uncertainty and noise," in *RSS Workshop on Model Learning for Human-Robot Communication*, 2016.

[10] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective," in *HRI 2012*. ACM, 2012, pp. 391–398.

[11] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *NIPS*, 2017, pp. 4299–4307.

[12] A. Wilson, A. Fern, and P. Tadepalli, "A bayesian approach for policy learning from trajectory preference queries," in *NIPS*.

[13] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

[14] A. Blidaru, S. L. Smith, and D. Kulic, "Assessing user specifications for robot task planning," in *RO-MAN*, 2018.

[15] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 4th ed. Springer Publishing Company, Inc., 2007.

[16] K. G. Jamieson and R. Nowak, "Active ranking using pairwise comparisons," in *NIPS*, 2011, pp. 2240–2248.

[17] L. Wasserman, *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated, 2010.

[18] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American statistical association*, vol. 58, no. 301, pp. 13–30, 1963.

[19] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.