Learning Control Sets for Lattice Planners from User Preferences

Alexander Botros*, Nils Wilde*, and Stephen L. Smith

Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, ON, Canada
{alexander.botros,nils.wilde,stephen.smith}@uwaterloo.ca

Abstract. This work investigates the design of a motion planner that can capture user preferences. In detail, we generate a sparse control set for a lattice planner which closely follows the preferences of a user. Given a set of demonstrated trajectories from a single user, we estimate user preferences based on a weighted sum of trajectory features. We then optimize a set of connections in the lattice of given size for the user cost function. The restricted number of connections limits the branching factor, ensuring strong performance during subsequent motion planning. Further, every trajectory in the control set reflects the learned user preference while the sub-optimality due to the size restriction is minimized. We show that this problem is optimally solved by applying a separation principle: First, we find the best estimate of the user cost function given the data, then an optimal control set is computed given that estimate. We evaluate our work in a simulation for an autonomous robot in a four-dimensional spatiotemporal lattice and show that the proposed approach is suitable to replicate the demonstrated behaviour while enjoying substantially increased performance.

Keywords: Motion and Path Planning, Mobile Robots

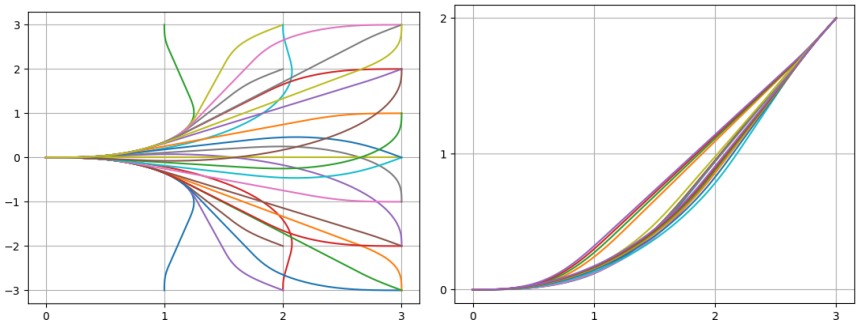
1 Introduction

In this paper we investigate the design of a motion planner that represents user preferences. We focus on lattice planners, where the problem becomes one of learning a control set. Lattice planners are widely used in autonomous driving [1–5]. Unlike commonly used probabilistic motion planners such as PRM$^{(*)}$, RRT$^{(*)}$ or FMT [6], lattices consist of a *regular* discretization of a robot's state space while ensuring kinematic and dynamic constraints on motions. Lattice planners pre-compute feasible trajectories between a discrete set of robot states. A pair of states together with a trajectory between the states is called a *motion primitive*; a collection of motion primitives forms a *control set*. Thus, the robot's dynamics

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

*Alexander Botros and Nils Wilde contributed equally.

do not need to be accounted for during the actual path planning. In Figure 1 we illustrate an example of a state lattice using clothoid trajectories [7], as well as a set of different trajectories between two fixed states obtained by varying the maximum curvature and maximum derivative of curvature. There are two



(a) Clothoid trajectory control set. (b) Sample clothoid trajectories.
Fig. 1: Examples of a lattice control set and different trajectories.

aspects to consider when designing a control set: 1) the set of states the motion primitives should connect (i.e., the endpoints of each primitive), and 2) the actual trajectory used to make the connections. We address the second aspect by estimating the trajectories that a user would take to each goal configuration in a discretization of the configuration space. Generally, the time to compute a motion plan and the quality of the resulting plan increase with the number of primitives [1, 8]. Thus we address the first aspect by selecting a *subset* of the estimated trajectories with a given size. To estimate the user trajectories, we consider that the quality of a plan is evaluated by a user who has preferences with respect to the robot's motion. In autonomous driving, some users may prefer the vehicle to drive more aggressively, making sharp turns and maintaining high speeds, while others may prefer slower speeds and smoother turns [9,10]. To find trajectories that reflect user preferences, we assume that users evaluate a robot's behaviour based on a weighted sum of features, similar to [9, 11].

The preferred behaviour of an autonomous robot often depends on situational context, like type of road or obstacles. Thus, a lattice planner based on trajectories that reflect the user preferences might only be applicable in certain scenarios. Complete planners that encompass many scenarios often use specialized lattices [9,12,13]. Our methodology is not limited to a specific scenario like driving on a highway or navigating in close proximity to an obstacle: for any of these scenarios we can learn a respective control set that captures the user preferences. Thus, we only consider global features such as trajectory length or maximum jerk, which are common in autonomous driving [9]. We learn the relative importance of features from demonstrations from a single user, i.e., estimate the weights and the user's cost function. The cost function is then used as an input to a trajectory planner, e.g., for a ground vehicle a Dubins path, clothoid [7]

or polynomial spiral [14], that computes the optimal trajectories between lattice states. Thus our approach is agnostic to the employed local planner.

A major restriction for the performance of lattice planners in high-dimensional states is the branching factor (the number of connections in the control set). Higher dimensional states are common in autonomous driving applications where the set of states can extend to (x, y) position, orientation, curvature an speed [4]. To improve runtime of the motion planner, we set an upper bound on the branching factor. Thus, there might not be a pre-computed trajectory to every state in the lattice; such states are reached by executing a sequence of trajectories. A lattice should then be constructed such that over all states for which no pre-computed direct trajectory is in the control set, the sub-optimality of the best trajectory sequence is minimized. This problem is closely related to the minimal t-spanning control set problem (MTSCSP) [1,8]. Here, we find an estimate of the user cost function and compute a lattice with a given maximum branching factor best suited for the user. Restricting the size of the control set vastly improves planning time but at the expense of path quality.

1.1 Related Work

Closely related to our work, [12] and [9] learn a human driving style from demonstrations, using a linear cost function weighting pre-defined features, and then generate paths using a graph or lattice. In contrast to our approach, both earlier approaches do not compute a new lattice for a learned user cost function, but instead search over a given graph or lattice, using the updated cost function. Furthermore, they consider local features that describe the relation of the vehicle to the environment, together with global features that are independent of the environment. In contrast, we only consider global features describing the driving style, i.e., the trade-off between travel time and passenger comfort, given a situation. Instead of keeping suboptimal trajectories between lattice points, we recompute all trajectories given a learned preference and then compute a sparse control set for the lattice. Thus, during the motion planning, the control set contains only trajectories that are optimal for the learned user preferences.

Research in human robot interaction studies how the behaviour of autonomous robots can be shaped to satisfy the preferences of users. Similar to work in learning from demonstration [15] or active preference learning [11, 16] we employ a user cost function that puts weights on a set of features. Given a fixed set of features, the user's cost function is learned by estimating the weights on the features. In [10] tunable parameters are introduced to a motion planner suited for urban driving while the work of [17] proposes a set of features to identify different driving behaviours of human drivers.

State lattices have found increasing popularity in motion planning for general problems in a 2D environment [1, 2], in autonomous driving [4, 5, 18] as well as high-dimensional systems [19, 20]. The authors of [1] proposed the problem of computing approximated control sets, i.e., smaller control sets that do not contain a direct motion primitive to every state. These states can still be reached via concatenations of the motions in the approximated set. The approximation

is characterized by the cost ratio between the path using concatenations and the direct motion. In [8] we proved NP-hardness for finding such an approximation with a given cost ratio, and introduced a mixed integer-linear program (MILP) formulation. However, [8] does not consider costs other than known path length, and does not account for user preferences when designing trajectories. In this paper we show that such an approximation yields an optimal solution for finding a control set of constrained size that adheres to user preferences on the motions.

1.2 Contributions

We introduce a novel methodology for designing a state lattice planner that considers user preferences. We pose the problem of simultaneously learning trajectories and a connection set of fixed size given demonstrations from a single user. The major contribution of this work is showing that an optimal solution to this problem can be found by applying a separation principle: First, we find the best estimate of the user preferences given the data, which define trajectories for all states in the lattice. Then we compute a connection set of given size that minimizes the error compared to using all connections in the lattice. We show that the proposed approach minimizes the maximum relative error of the expected optimal trajectories. Finally, we demonstrate the performance in simulations where the average cost of the paths computed with the learned control set is 1.4 of the optimum. Moreover, we obtain a substantial planning time speedup of more than factor 10 when using the learned connection set compared to planning with the entire lattice.

2 Problem Statement

2.1 Preliminaries

Graph Theory Following [21], a graph is an ordered pair $G = (V, E)$, where V is a set of vertices and E is a set of edges. In a weighted graph $G = (V, E, c)$ a real valued function associates a cost to each edge of the graph: $c : E \to \mathbb{R}$. We define a path $P_{s,g}$ between two vertices s and g in V as a sequence of edges $P_{s,g} = (e_0, \dots, e_k)$ where $e_i = (v_i, v_{i+1})$, $v_0 = s, v_{k+1} = g$, and $v_j \neq v_m$ for all $j, m \in \{0, \dots, k\}$. The cost of a path is defined as $c(P_{s,g}) = \sum_{e \in P_{s,g}} c(e)$.

Spatio-Temporal Lattice Planner Following [2], Given the state space of a mobile robot X, let $V \subset X$ denote a regularly spaced, finite subset of robot states, also called lattice states, and let $s \in V$ denote an arbitrary starting state. Further, $\mathcal{B} = \{(s, j) : j \in V\}$ is the set of tuples of s and all vertices $j \in V$. We call any $\mathcal{E} \subseteq \mathcal{B}$ a *connection set*. Each tuple can be thought of as a start-goal pair of states in the configuration space. Let T denote a set containing a unique trajectory from s to j for each $(s, j) \in \mathcal{B}$. The key idea is that connections in $\mathcal{B}$ – together with their associated trajectories – can also be applied on other vertices $j \in V$ to reach another state i in X. By pre-computing trajectories from

s to j for all $j \in V$, we can construct trajectories from s to i via successive valid concatenations [8] of the trajectories in T.

Given a set of connections $\mathcal{E} \subseteq \mathcal{B}$ the tuple $(\mathcal{E}, T)$ is called a *control set* of the lattice. For any $b = (s, j) \in \mathcal{E}$, and trajectory $T_b \in T$ from s to j, we call the tuple $(b, T_b) \in \mathcal{E} \times T$ a *motion primitive*. For a given control set $(\mathcal{E}, T)$ with $\mathcal{E} \subseteq \mathcal{B}$, we may define a weighted, directed graph $G_T^{\mathcal{E}} = (V, E, c)$. The edges E are all those pairs (i, j) such that there exists a connection $b = (s, k) \in \mathcal{E}$ that takes i to j. The cost c of the edge (i, j) is the cost of the trajectory from s to k. This cost is assumed to be an *almost-metric* (that is, it satisfies all properties of a metric except for the symmetry property). The weighted, directed graph $G_T^{\mathcal{E}}$ is called a *state lattice*. Similar to $G_T^{\mathcal{E}}$ we can define a graph $G_T^{\mathcal{B}} = (V, B, c)$ where B are all edges defined by connections $b \in \mathcal{B}$.

Minimal Connection Sets This section reviews the work presented in [2,8]. Given $G_T^{\mathcal{E}} = (V, E, c)$ let $c_j^{\mathcal{E}}$ be the cost of the minimal-cost path from s to vertex j in $G_T^{\mathcal{E}}$. This definition together with the assumption that c is an almost metric (and therefore satisfies the triangle inequality) implies that $c_j^{\mathcal{B}}$ is the cost of the direct trajectory s to j, i.e., the trajectory computed without concatenation. To evaluate the quality of a control set $\mathcal{E}$, we use the t-error of defined here:

Definition 1 (t-error). *Given a lattice graph $G_T^{\mathcal{B}} = (V, B, c)$, the* t-error *of a subset $\mathcal{E} \subseteq \mathcal{B}$ is given by*

$$\tau(\mathcal{E}) = \max_{j \in V} \left(c_j^{\mathcal{E}} / c_j^{\mathcal{B}} \right). \tag{1}$$

For each $j \in V$, the value $c_j^{\mathcal{E}} / c_j^{\mathcal{B}}$ represents the ratio of the cost-minimal path using connections only in $\mathcal{E}$ to the cost of the direct trajectory from s to j.

The *Minimum t-Spanning Control Set Problem* (MTSCSP) is formulated as follows: Given a state lattice $G_T^{\mathcal{B}}$, compute a set $\mathcal{E} \subseteq \mathcal{B}$ of minimal size such that $\tau(\mathcal{E}) \leq t$. This problem is NP-complete.

2.2 Problem Formulation

In the preliminaries, we discussed the problem of computing a minimal t-spanning control set of a lattice $G_T^{\mathcal{B}} = (V, B, c)$. However, the formulation of this problem requires that the set of trajectories T, and thus the costs c are known. Different users might have individual preferences for trajectories, which can be described by a respective cost function c. We model users who evaluate a trajectory T_j from s to j as a weighted sum of features $\boldsymbol{\phi}(T_j) = \left[\phi_1(T_j)\ \phi_2(T_j)\ \dots\ \phi_n(T_j)\right]^T$:

$$c(T_j, \boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{\phi}(T_j), \tag{2}$$

where $\boldsymbol{w}$ is a vector of weights $\left[w_1\ w_2\ \dots\ w_n\right]$. The features can represent properties such as travel time, lateral acceleration, maximum jerk, etc. Similar user cost functions have been used in [11,15,16,22]. Without loss of generality, we assume that $\boldsymbol{w} \in [0, 1]^n$. Given a user weight $\boldsymbol{w}$, we denote the cost of the optimal

trajectory from s to j with respect to the weights $\boldsymbol{w}$ as

$$c_j^*(\boldsymbol{w}) = \min_{T_j} \ \boldsymbol{w} \cdot \phi(T_j). \tag{3}$$

For a given set of weights $\boldsymbol{w} \in [0,1]^n$, and a given set of connections $\mathcal{E} \subseteq \mathcal{B}$, let $G^{\mathcal{E},\boldsymbol{w}} = (V, E, c^*)$. Here, E is the set of all pairs (i,j) such that there exists a connection $b = (s,k) \in \mathcal{E}$ taking i to j, and $c^*(i,j) = c_k^*(\boldsymbol{w})$. For any weight $\boldsymbol{w} \in [0,1]^n$, let $P_j^{\mathcal{E}}$ be a path from s to j on graph $G^{\mathcal{E},\boldsymbol{w}}$. For every edge e in a path $P_j^{\mathcal{E}}$ there is an associated connection $b \in \mathcal{E}$, and for every connection $b \in \mathcal{E}$, there is an associated trajectory $T_b \in T$. Thus a path $P_j^{\mathcal{E}}$ from s to j defines a trajectory from s to j which is the sequence of trajectories associated with each edge in the path. Let $(T_1, T_2, \dots)$ be the sequence of trajectories associated with the edges $(e_1, e_2, \dots)$ in a path $P_j^{\mathcal{E}}$. We extend the cost function c for trajectories to a cost function u of paths: For each feature ϕ_l there exists a feature function

$$f_l(P_j^{\mathcal{E}}) = f_l(\{\phi_l(T_1), \phi_l(T_2), \dots\}). \tag{4}$$

Here f_l depends of the trajectories associated with the edges in $e_1, e_2, \cdots \in P_j^{\mathcal{E}}$. Further, we only consider functions $f_l(\cdot)$ that are monotone, i.e., $f_l(P^1) \leq f_l(P^2)$ if $P^1 \subseteq P^2$. For example, let ϕ_l be the length of a trajectory, then f_l describing the length of a path sums over all $\phi_l(T_1), \phi_l(T_2), \dots$. Furthermore, for other features we might require a non-linear mapping, e.g., the maximum curvature of a path is the maximum curvature among its trajectories. The user cost function of a path is then given by the weighted sum of all features

$$u(P_j^{\mathcal{E}}, \boldsymbol{w}) = w_1 f_1(P_j^{\mathcal{E}}) + \cdots + w_n f_n(P_j^{\mathcal{E}}). \tag{5}$$

We notice that u is a generalization of the cost function c, i.e., c evaluates a single trajectory while u is a function of a set of trajectories which form a path. The user cost function u is similar to a reward function in reinforcement learning. However, it is challenging for users – especially non-experts – to specify weights for such a reward function [11]. Summarizing all weights in a row vector $\boldsymbol{w}$ and all features in a column vector $\boldsymbol{f}(P_j^{\mathcal{E}})$ allows us to write $u(P_j^{\mathcal{E}}, \boldsymbol{w}) = \boldsymbol{w}\boldsymbol{f}(P_j^{\mathcal{E}})$. Given weights $\boldsymbol{w}$, we define the optimal path as

$$P_j^{\mathcal{E},\boldsymbol{w}} = \arg\min_{P_j^{\mathcal{E}}} u(P_j^{\mathcal{E}}, \boldsymbol{w}). \tag{6}$$

Observe that this definition implies that

$$u(P_j^{\mathcal{B},\boldsymbol{w}}, \boldsymbol{w}) = c_j^*(\boldsymbol{w}). \tag{7}$$

Indeed, for connection set $\mathcal{B}$ there exists a direct connection taking s to j for all $j \in V$, and the cost of the minimal trajectory given $\boldsymbol{w}$ is given by $c_j^*(\boldsymbol{w})$.

Given a set of connections $\mathcal{E}$, and weights $\boldsymbol{w}$, let $T = \{T_b \ : \ b \in \mathcal{E}\}$ where T_b solves (3) for each $b \in \mathcal{E}$. That is, T is the set of minimal cost trajectories with respect to $\boldsymbol{w}$ for each connection in $\mathcal{E}$. We observe that a control set $(\mathcal{E}, T)$

can thus be determined by the tuple $(\mathcal{E}, \boldsymbol{w})$. For the remainder of this paper, we focus on computing the tuple $(\mathcal{E}, \boldsymbol{w})$. Based on our cost function, the user optimal behavior between all connections in $\mathcal{B}$ can be described by a control set $(\mathcal{B}, \boldsymbol{w}^*)$ where $\boldsymbol{w}^*$ denotes the *true user weights*. Users cannot provide $\boldsymbol{w}^*$ directly, and we learn about $\boldsymbol{w}^*$ from demonstrations. The true user weights may be dependent on the situation (e.g. highway vs city driving). This work focuses on a single situation but could be extended to account for multiple situations with the result being several control sets, one for each situation. For a given situation, we treat $\boldsymbol{w}^*$ as a hidden parameter which we have to estimate. Consider a path $P_j^{\mathcal{E},\boldsymbol{w}_1}$ that is optimal for a control set $(\mathcal{E}, \boldsymbol{w}_1)$, and a second weight $\boldsymbol{w}_2$. The cost of path $P_j^{\mathcal{E},\boldsymbol{w}_1}$ can be *evaluated* by $\boldsymbol{w}_2$, which we write as:

$$u_j(\mathcal{E}, \boldsymbol{w}_1 | \boldsymbol{w}_2) = \boldsymbol{w}_2 \cdot \boldsymbol{f}(P_j^{\mathcal{E},\boldsymbol{w}_1}) = u(P_j^{\mathcal{E},\boldsymbol{w}_1}, \boldsymbol{w}_2). \tag{8}$$

We read this as *the cost to j using control set $(\mathcal{E}, \boldsymbol{w}_1)$, evaluated by weights $\boldsymbol{w}_2$, i.e., the cost of the path that is optimal given weights $\boldsymbol{w}_1$, for a user whose weights are $\boldsymbol{w}_2$.* This concept is similar to the regret of optimization problems [23]. Based on this notation, we can now pose the main problem statement:

Problem 1 (User control set)**.** Given a finite set of lattice states, V, hidden user preferences $\boldsymbol{w}^*$, and a budget on the number of allowable connections, $k \in \mathbb{Z}_{>0}$, find a control set $(\mathcal{E}, \boldsymbol{w})$, with $\mathcal{E} \subseteq \mathcal{B}, \boldsymbol{w} \in [0,1]^n$ such that

$$\begin{aligned}(\mathcal{E}, \boldsymbol{w}) = \arg\min_{\mathcal{E}', \boldsymbol{w}'} \max_{j \in V} \frac{u_j(\mathcal{E}', \boldsymbol{w}' | \boldsymbol{w}^*)}{c_j^*(\boldsymbol{w}^*)} \\ s.t.\ |\mathcal{E}'| \leq k,\ \boldsymbol{w}' \in [0,1]^n.\end{aligned} \tag{9}$$

The control set $(\mathcal{E}, \boldsymbol{w})$ minimizes the maximum ratio of path costs *evaluated* by $\boldsymbol{w}^*$ to the cost of the optimal direct trajectory to j, given a budget of k connections. In essence $(\mathcal{E}, \boldsymbol{w})$ is the control set with the best t-error and is therefore the most robust control.

3 Approach

Problem 1 consists determining a control set $(\mathcal{E}, \boldsymbol{w})$ that minimizes the maximum ratio between path costs and optimal cost. We introduce a model for how users provide demonstrations and then analyse how the unknown parameter $\boldsymbol{w}^*$ in equation (9) can be substituted by an estimate of the user weights given data. We show that the optimal solution is a pair $(\mathcal{E}, \boldsymbol{w})$ where $\boldsymbol{w}$ is the best available estimate of $\boldsymbol{w}^*$ and $\mathcal{E}$ can be computed via a mixed integer linear program. This allows for a simple, yet effective method for solving Problem 1.

3.1 User model

We consider a user with a preference $\boldsymbol{w}^*$. Let d be a demonstrated trajectory from a start state s to a goal state $j \in X$. We do not require j to be a lattice point

as we can still evaluate features $\boldsymbol{f}(d)$ and thus assign a cost $u_j(d, \boldsymbol{w})$ as in (5). We denote $d^{\boldsymbol{w}^*}$ the minimal-cost demonstration from s to j given weights $\boldsymbol{w}^*$. Users cannot perfectly demonstrate $d^{\boldsymbol{w}^*}$. Thus we use the features of demonstrations to formulate a probabilistic user model:

Assumption 1 (User Model)**.** A user with a preference $\boldsymbol{w}^*$ provides a demonstration d from s to j with features $\boldsymbol{f}(d)$ where the density $p(\boldsymbol{f}(d)|\boldsymbol{w}^*)$ is:

$$p(\boldsymbol{f}(d)|\boldsymbol{w}^*) \sim \mathcal{N}(\boldsymbol{f}(d^{\boldsymbol{w}^*}), \boldsymbol{\sigma}). \tag{10}$$

Similar user models have been used in [19, 22]. Thus, users provide demonstrations such that the features follow a normal distribution centred at the features of the optimal demonstration. That is the user demonstrations are unbiased and, given a large enough data set, the average features of demonstrations equal the optimal features. Following (5), two demonstrations with equal features have the same cost for any user and thus are indistinguishable with respect to the cost function. Hence, we consider only features of demonstrations. Let $\mathcal{D} = (d_1, d_2, \dots)$ be a sequence of demonstrations. We then find the conditional expectation of $\boldsymbol{w}$ given $\mathcal{D}$ by taking the Bayesian posterior:

$$\mathbb{E}_{\boldsymbol{w}}[\boldsymbol{w}|\mathcal{D}] = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} \mathbb{E}[\boldsymbol{w}|\boldsymbol{f}(d)]p(\boldsymbol{f}(d)) = \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} \int_{\boldsymbol{w}\in[0,1]^n} \boldsymbol{w}\ p(\boldsymbol{f}(d)|\boldsymbol{w})\ p(\boldsymbol{w}) d\boldsymbol{w}. \tag{11}$$

Finally, we can approximate the integral by summing over a set of N samples:

$$\mathbb{E}_{\boldsymbol{w}}[\boldsymbol{w}|\mathcal{D}] \approx \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{w}_i\ p(\boldsymbol{f}(d)|\boldsymbol{w}_i)\ p(\boldsymbol{w}_i). \tag{12}$$

3.2 Estimation of the loss function

We now use the user model to find a solution to Problem 1, given a set of user demonstrations $\mathcal{D}$. We consider two approaches. The first is taking the conditional expectation over equation (9), given $\mathcal{D}$:

$$\begin{aligned}(\bar{\mathcal{E}}, \bar{\boldsymbol{w}}) = \underset{\mathcal{E}', \boldsymbol{w}'}{\arg\min}\ \mathbb{E}_{\boldsymbol{w}}\left[\max_{j\in V}\ \frac{u_j(\mathcal{E}', \boldsymbol{w}'|\boldsymbol{w})}{c_j^*(\boldsymbol{w})}\Big|\mathcal{D}\right] \\ s.t.\ |\mathcal{E}'| \le k,\ \boldsymbol{w}' \in [0,1]^n.\end{aligned} \tag{13}$$

The second approach is to use the expectation $\hat{\boldsymbol{w}} = \mathbb{E}[\boldsymbol{w}|\mathcal{D}]$ to compute $(\bar{\mathcal{E}}, \bar{\boldsymbol{w}})$, also known as the plug-in estimator [24]:

$$\begin{aligned}(\bar{\mathcal{E}}, \bar{\boldsymbol{w}}) = \underset{\mathcal{E}', \boldsymbol{w}'}{\arg\min}\ \max_{j\in V}\ \frac{u_j(\mathcal{E}', \boldsymbol{w}'|\hat{\boldsymbol{w}})}{c_j^*(\hat{\boldsymbol{w}})} \\ s.t.\ |\mathcal{E}'| \le k,\ \boldsymbol{w}' \in [0,1]^n.\end{aligned} \tag{14}$$

While it is an approximation, it approaches the desired Problem 1 as the number of demonstrations goes to infinity. Thus, this work focuses on solving (14).

3.3 Main Results

In this section, we present the main theorem of this paper that proposes a solution to the minimization problem in (14). The high-level idea is: given demonstrations $\mathcal{D}$, the expected user weight $\hat{\boldsymbol{w}}$ is computed. This user weight is used to calculate trajectories that minimize the user cost for all connections in $\mathcal{B}$. Finally, a set $\mathcal{E} \subseteq \mathcal{B}$ of size $\leq k$ is found to produce a control set is $(\mathcal{E}, \hat{w})$. In order to illustrate how the connection set $\mathcal{E}$ is computed, we present a variant of the MTSCSP from [2,8] that is used in the main results of this paper.

Problem 2 (Minimum k-spanning Connection set Problem (MKSCSP))**.** Given a tuple $(V, \mathcal{B}, c)$, and an integer $k > 0$, compute a set $\mathcal{E}$ such that

$$\mathcal{E} = \arg \min_{\substack{\mathcal{E}' \subseteq \mathcal{B} \\ |\mathcal{E}| \leq k}} \tau(\mathcal{E}'), \tag{15}$$

where $\tau(\mathcal{E})$ is defined in (1) with $c(j) = c_j^{\mathcal{B}}$ for all $j \in V$. If $\mathcal{E}$ solves (15), and $\tau(\mathcal{E}) < \infty$ we say that $\mathcal{E}$ is a *minimal k-spanning connection set* (MKSCS).

We make an observation about the cost u function to prepare our main result.

Observation 1 (Weight Choice)**.** For any set of connections $\mathcal{E}$, any vertex $j \in V$, and any pair of weights $\boldsymbol{w}, \boldsymbol{w}' \in [0,1]^n$, it must hold that

$$u_j(\mathcal{E}, \boldsymbol{w} | \boldsymbol{w}) \leq u_j(\mathcal{E}, \boldsymbol{w}' | \boldsymbol{w}). \tag{16}$$

Indeed, from the definition of paths $P_j^{B,\boldsymbol{w}}$ and user costs in (6), and the definition of user costs *evaluated* by other weights in (8), we observe that

$$u_j(\mathcal{E}, \boldsymbol{w} | \boldsymbol{w}) = u(P_j^{\mathcal{E},\boldsymbol{w}}, \boldsymbol{w}) \leq u(P_j^{\mathcal{E},\boldsymbol{w}'}, \boldsymbol{w}) = u_j(\mathcal{E}, \boldsymbol{w}' | \boldsymbol{w}). \tag{17}$$

Theorem 1 (Problem Solution). *If the tuple $(\bar{\mathcal{E}}, \bar{\boldsymbol{w}})$ is a solution to minimization problem* (14)*, then $\bar{\boldsymbol{w}} = \hat{\boldsymbol{w}}$, and $\bar{\mathcal{E}}$ is a MKSCS of the lattice whose costs are computed using weights $\hat{\boldsymbol{w}}$ (i.e., the lattice $G^{\mathcal{B},\hat{\boldsymbol{w}}}$).*

Proof. Let $\hat{\mathcal{E}}$ denote a MKSCS of the lattice $G^{\mathcal{B},\hat{\boldsymbol{w}}}$. Note that Observation 1 implies that the optimal value for $\bar{\boldsymbol{w}}$ in equation (14) is given by $\hat{\boldsymbol{w}}$ for any set of connections $\mathcal{E}'$. Indeed, for any weights $\boldsymbol{w}' \in [0,1]^n$, equation (16) implies that $u_j(\mathcal{E}', \boldsymbol{w}' | \hat{\boldsymbol{w}}) \geq u_j(\mathcal{E}', \hat{\boldsymbol{w}} | \hat{\boldsymbol{w}})$. Thus, for any $\mathcal{E}'$, the pair $(\mathcal{E}', \boldsymbol{w}')$ must result in paths whose cost is at least $(\mathcal{E}', \hat{\boldsymbol{w}})$. This allows us to simplify equation (14) to

$$\begin{aligned} \bar{\mathcal{E}} = \arg\min_{\mathcal{E}'} \max_{j \in V} \frac{u_j(\mathcal{E}', \hat{\boldsymbol{w}} | \hat{\boldsymbol{w}})}{c_j^*(\hat{\boldsymbol{w}})}. \\ s.t.\ |\mathcal{E}'| \leq k. \end{aligned} \tag{18}$$

We will now show that $\hat{\mathcal{E}}$ solves (18). Observe that the t-error for any set $\mathcal{E}'$ given the lattice $G^{\mathcal{B},\hat{\boldsymbol{w}}}$ is exactly $u_j(\mathcal{E}', \hat{\boldsymbol{w}} | \hat{\boldsymbol{w}}) / c_j^*(\hat{\boldsymbol{w}})$ which appears in the minimization

in (18). Therefore, it must hold that if $\hat{t} = \tau(\hat{\mathcal{E}})$ given the lattice $G^{\mathcal{B},\hat{\boldsymbol{w}}}$, then for any other set of primitives $\bar{\mathcal{E}} \subseteq \mathcal{B}$ such that $|\bar{\mathcal{E}}| \leq k$ we have

$$\tau(\hat{\mathcal{E}}) \leq \tau(\bar{\mathcal{E}}). \tag{19}$$

This follows from the assumption that $\hat{\mathcal{E}}$ is a MKSCS for the lattice $G^{\mathcal{B},\hat{\boldsymbol{w}}}$. Thus $\bar{\mathcal{E}}$ solves the minimization problem (18). Finally, let $\bar{\mathcal{E}}$ be any solution to (18). Then, it must hold that $\tau(\bar{\mathcal{E}}) = \hat{t}$, implying $\bar{\mathcal{E}}$ is also a MKSCS of the lattice $G^{\mathcal{B},\hat{\boldsymbol{w}}}$. Therefore, we have that $\boldsymbol{w}'$ is a weight that solves (14) if and only if $\boldsymbol{w}' = \hat{\boldsymbol{w}}$. Given that $\boldsymbol{w}' = \hat{\boldsymbol{w}}$, allows us to reduce (14) to (18) which is solved by a connection set $\bar{\mathcal{E}}$, if and only $\bar{\mathcal{E}}$ is a MKSCS of the lattice $G^{\mathcal{B},\hat{\boldsymbol{w}}}$. □

Corollary 1. *If the control set $(\mathcal{E}, \boldsymbol{w})$ is a solution to equation (9), then $\boldsymbol{w} = \boldsymbol{w}^*$, and $\mathcal{E}$ is a MKSCS on the lattice $G^{\mathcal{B},\boldsymbol{w}^*}$.*

The proof of Corollary (1) follows closely the proof of Theorem 1, and is omitted. Theorem 1 shows that $(\hat{\mathcal{E}}, \hat{\boldsymbol{w}})$ minimizes the t-error of a control set given the lattice whose weights are the expected weight given $\mathcal{D}$. The next theorem extends this result to a lattice whose trajectory costs are given by their expected costs, given $\mathcal{D}$, if the features $f_i, i \in \{1, \dots\}$ are piece-wise continuous functions of $\boldsymbol{w}$. We say that a multivariate function $f_i(\boldsymbol{w})$ is piece-wise continuous on the compact set $[0,1]^n$ if for any two weight vectors $\boldsymbol{w}_1, \boldsymbol{w}_2 \in [0,1]^n$, the function f_i is piece-wise continuous on the line-segment connecting $\boldsymbol{w}_1, \boldsymbol{w}_2$.

Theorem 2 (Solution for the Expected Cost Lattice). *Let $(\hat{\mathcal{E}}, \hat{\boldsymbol{w}})$ be a solution to problem (14). If the features f_i are piece-wise continuous functions of $\boldsymbol{w}$ for all $i \in \{1, \dots, n\}$ then for all $j \in V$ it holds that*

$$(\hat{\mathcal{E}}, \hat{\boldsymbol{w}}) = \arg\min_{\mathcal{E}', \boldsymbol{w}'} \max_{j \in V} \frac{\mathbb{E}_{\boldsymbol{w}}[u_j(\mathcal{E}', \boldsymbol{w}'|\boldsymbol{w})|\mathcal{D}]}{\mathbb{E}_{\boldsymbol{w}}[c_j^*(\boldsymbol{w})|\mathcal{D}]}. \tag{20}$$

That is, the control set $(\hat{\mathcal{E}}, \hat{\boldsymbol{w}})$ minimizes the t-error relative to the expected cost of the optimal trajectory to each $j \in V$ given $\mathcal{D}$.

Proof. We first show that for any fixed $\mathcal{E}' \subseteq \mathcal{B}$ and $j \in V$, the function $u_j(\mathcal{E}', \boldsymbol{w}|\boldsymbol{w})$ is a piece-wise linear function of $\boldsymbol{w}$. If we fix the first argument $\boldsymbol{w}'$, the cost function $u_j(\mathcal{E}', \boldsymbol{w}'|\boldsymbol{w})$ becomes linear in $\boldsymbol{w}$. Indeed, from (5) and (8), we have $u_j(\mathcal{E}', \boldsymbol{w}'|\boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{f}(P_j^{\mathcal{E}', \boldsymbol{w}'})$, where $\boldsymbol{f}(P_j^{\mathcal{E}', \boldsymbol{w}'})$ is constant for a fixed control set $(\mathcal{E}', \boldsymbol{w}')$. Because the features $\boldsymbol{f}$ are assumed to be piece-wise continuous, we can conclude from (8) that $u(\mathcal{E}', \boldsymbol{w}|\boldsymbol{w})$ is also piece-wise continuous. Let $\epsilon > 0, \boldsymbol{w}_1, \boldsymbol{w}_2$ be such that $u(\mathcal{E}', \boldsymbol{w}|\boldsymbol{w})$ is continuous on a line-segment

$$L = \{\lambda_1 \boldsymbol{w}_1 + \lambda_2 \boldsymbol{w}_2 \ : \ \lambda_1 \in [-\epsilon, 1+\epsilon], \ \lambda_2 = (1-\lambda_1)\}. \tag{21}$$

That is, $u(\mathcal{E}', \boldsymbol{w}|\boldsymbol{w})$ is continuous on a line segment containing and extending a small distance depending on ϵ past $\boldsymbol{w}_1, \boldsymbol{w}_2$. Observe that the linearity of $u(\mathcal{E}', \boldsymbol{w}'|\boldsymbol{w})$ for any fixed weights $\boldsymbol{w}'$, together with Observation 1 imply that

$$\begin{aligned} u_j(\mathcal{E}', &\lambda_1 \boldsymbol{w}_1 + \lambda_2 \boldsymbol{w}_2 | \lambda_1 \boldsymbol{w}_1 + \lambda_2 \boldsymbol{w}_2) \\ &= \lambda_1 u_j(\mathcal{E}', \lambda_1 \boldsymbol{w}_1 + \lambda_2 \boldsymbol{w}_2 | \boldsymbol{w}_1) + \lambda_2 u_j(\mathcal{E}', \lambda_1 \boldsymbol{w}_1 + \lambda_2 \boldsymbol{w}_2 | \boldsymbol{w}_2) \\ &\geq \lambda_1 u_j(\mathcal{E}', \boldsymbol{w}_1 | \boldsymbol{w}_1) + \lambda_2 u_j(\mathcal{E}', \boldsymbol{w}_2 | \boldsymbol{w}_2). \end{aligned} \tag{22}$$

Thus, the function u_j lies above the line passing through $u_j(\mathcal{E}', \boldsymbol{w}_1|\boldsymbol{w}_1)$, and $u_j(\mathcal{E}', \boldsymbol{w}_2|\boldsymbol{w}_2)$. Since this holds for all $\boldsymbol{w}_1', \boldsymbol{w}_2' \in L$, and $u_j(\mathcal{E}', \boldsymbol{w}|\boldsymbol{w})$ is continuous on the line connecting $\boldsymbol{w}_1', \boldsymbol{w}_2'$, it must be linear there. Thus, if $\boldsymbol{f}(\boldsymbol{w})$ is continuous on any line segment, then $u_j(\mathcal{E}', \boldsymbol{w}|\boldsymbol{w})$ is linear there. Were the values of λ_1, λ_2 in (22) restricted to $[0, 1]$, it would only imply concavity, but (22) holds for all $\lambda_1, \lambda_2 \in \mathbb{R}_{\geq 0}$, implying piece-wise linearity. By the linearity of the cost function, $u_j(\mathcal{E}', \boldsymbol{w}'|\boldsymbol{w})$ in $\boldsymbol{w}$ for fixed $\boldsymbol{w}'$, we have $\mathbb{E}_{\boldsymbol{w}}[u_j(\mathcal{E}', \boldsymbol{w}'|\boldsymbol{w})|\mathcal{D}] = u_j(\mathcal{E}', \boldsymbol{w}'|\hat{\boldsymbol{w}})$. By the linearity of $u_j(\mathcal{E}', \boldsymbol{w}|\boldsymbol{w})$ in $\boldsymbol{w}$ for any $\mathcal{E}' \subseteq \mathcal{B}$, and (7) we have $\mathbb{E}_{\boldsymbol{w}}[c_j^*(w)|\mathcal{D}] = c_j^*(\hat{\boldsymbol{w}})$. Thus the minimization problem (20) reduces to (14), the solution to which is given by $(\hat{\mathcal{E}}, \boldsymbol{w})$ by Theorem 1. □

3.4 Computational Complexity

In this section we prove the NP-completeness of the decision version of the minimization (9). We begin by stating the decision version of Problem 2.

Problem 3 (Decision version of MKSCSP)**.** Given tuple $(V, \mathcal{B}, c)$, integer $k > 0$, and a real number $t \geq 1$, does there exist a set $\mathcal{E} \subseteq \mathcal{B}$ with $|\mathcal{E}| \leq k$ and $\tau(\mathcal{E}) \leq t$?

Theorem 3 (NP-completeness). *Problem 3 is NP-complete.*

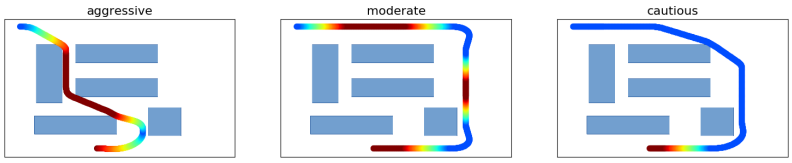
Proof. The MTSCSP in the preliminary section was shown to be NP-complete in [8]. Its decition version is: *Given a lattice $G^{\mathcal{B},\boldsymbol{w}}$, a real number $t \geq 1$ and an integer $k > 0$, does there exist a set $\mathcal{E} \subseteq \mathcal{B}$ where $\tau(\mathcal{E}) \leq t$ and $|\mathcal{E}| \leq k$?* We reduce the MTSCSP decision problem to Problem 3. Given a lattice $G^{\mathcal{B},\boldsymbol{w}}$, and $t \geq 1, k > 0$, we construct an instance of the MKSCSP as follows: let $V = \{j : (s, j) \in \mathcal{B}\}$, and let $c(j) = c_j^*(\boldsymbol{w})$ for all $j \in V$. Then we observe that Problem 3 on this instance is identical to the decision version of the MTSCSP. Thus, Problem 3 is NP-hard because the MTSCSP decision problem is. Observe that a potential solution $\mathcal{E} \subseteq \mathcal{B}$ to Problem 3 can be verified in time polynomial $|V|$ by iterating over all vertices $j \in V$ to check $\tau(\mathcal{E}) \leq t$, and additionally checking that $|\mathcal{E}| \leq k$. Hence, Problem 3 is in NP and thus NP-complete. □

3.5 Computing an optimal control set

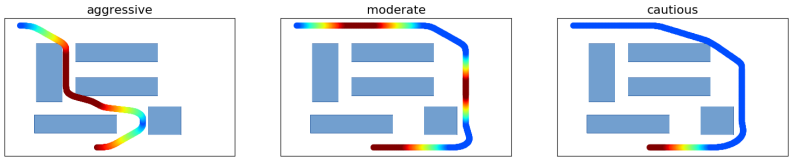
We now briefly summarize the proposed solution to problem (14). By Theorem 1, the solution is a tuple $(\hat{\mathcal{E}}, \hat{\boldsymbol{w}})$, where $\hat{\boldsymbol{w}}$ can be approximated using (12), and the MKSCS, $\hat{\mathcal{E}}$ is computed via a variant of the MTSCSP mixed integer program from [8]. This is done by adding the constraint that $|\hat{\mathcal{E}}| \leq k$, and changing the objective of the problem from minimizing $|\hat{\mathcal{E}}|$ to minimizing $\tau(\hat{\mathcal{E}})$.

4 Evaluation

We now demonstrate the performance of the proposed approach in simulations. We consider a known, static environment, discretized into four-dimensional states. Each state consists of a (x, y) position, an orientation θ and the current



(a) Demonstrated behaviour



(b) Learned behaviour

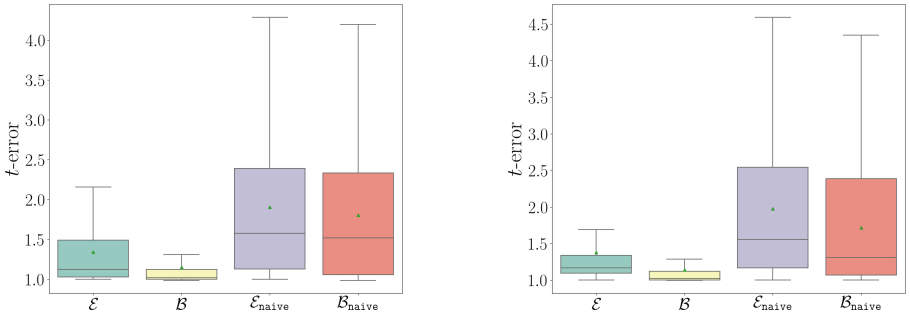
Fig. 2: Demonstrated and learned behaviour in the training environment. The color of the path indicates speed, where blue corresponds to slow and red corresponds to fast.

speed v. We considered 3 discrete values of speed, and 8 discrete headings (cardinal and ordinal). All used *environments* consisted of 15×15 grid of (x, y) positions and hence of $15 \times 15 \times 8 \times 3 = 5400$ states (vertices). The *lattice* had pre-computed trajectories for x-values between 0 and 4, y-values between -3 and 3, all 8 headings and all 3 speed values. Trajectories were computed using clothoids, a subset of the used lattice is illustrated in Figure 1. Each vertex has to be reached from a start for every discrete speed value and from a cardinal and ordinal orientation, yielding 6 different starts $(0, 0, j, v_i), j \in \{0, \pi/4\}, i \in \{1, 2, 3\}$. Thus, the connection set $\mathcal{B}$ for each start contains up to 672 connections.

We model users who evaluate trajectories based on three features: travel time, longitudinal acceleration and lateral acceleration. User demonstration were simulated by sampling features from the distribution in (10). We consider three preferences corresponding to different driving styles: An aggressive style which prefers short travel times, a cautious style that favours low accelerations, and a moderate style that balances the features more equally. For all user types we set the covariance in (10) to 0.1. We illustrate an example demonstration of each user in Figure 2a. Finally, the experiment comprises one training environment, shown in Figure 2 and two test environments.

4.1 Training Error

For each of the three driving style, we obtain three demonstrations for different start-goal pairs. We show that we can effectively learn user preferences despite this small number of demonstrations. We estimate $\hat{\boldsymbol{w}}$ for each user by computing the expectation in equation (12) using $N = 10$ samples and assuming a covariance of 0.05, i.e., we overestimate how accurately the user demonstration match their optimal trajectories. Finally, we run the experiment for minimal k-spanning connection sets of size 25, 50 and 100 over 40 trials each. Figure 2 shows one of the training demonstrations starting at the bottom at high speed and going to the top left corner. We compare the optimal paths for the three different users, together with the paths that were planned using the learned control set $(\hat{\mathcal{E}}, \hat{\boldsymbol{w}})$.



(a) Training Error (b) Test error

Fig. 3: The t-error for all connections $\mathcal{B}$ and the MKSCS connections $\mathcal{E}$ compared to a naive approach with $\mathcal{B}_{\texttt{naive}}$ and $\mathcal{E}_{\texttt{naive}}$, respectively.

While the shape of the paths is slightly different from the demonstrations, key characteristics of the user preferences are replicated. The aggressive user only breaks for sharp turns and immediately accelerates again. The moderate user avoids breaking too abruptly and thus cannot take the shortcut. Finally, the cautious user minimizes lateral and longitudinal acceleration and thus drives with minimal speed whenever possible. Figure 3a shows the t-error over all users and k values for the control set $(\hat{\mathcal{E}}, \hat{\boldsymbol{w}})$ together with the error of the control set using all connections $(\mathcal{B}, \hat{\boldsymbol{w}})$ in the lattice. We include a naive approach as a reference, where $\boldsymbol{w}_{\texttt{naive}}$ is sampled randomly and independent to the demonstrations. This naive approach serves as a baseline to illustrate the advantage of estimating driver behavior, i.e., the sensitivity of the cost function to user preferences $\boldsymbol{w}$. For this naive approach we show the error of a MKSCS $(\mathcal{E}_{\texttt{naive}}, \boldsymbol{w}_{\texttt{naive}})$ and of $(\mathcal{B}_{\texttt{naive}}, \boldsymbol{w}_{\texttt{naive}})$. We observe that $(\mathcal{B}, \hat{\boldsymbol{w}})$ achieves an average error of 1.16, with a median of 1.02. The MKSCS solution has a mean and median error of 1.34 and 1.13, respectively. Thus, the limited size of the control set leads to a mean cost that is 34% higher than the optimum, but for half of all cases the cost is at most 13% higher. In comparison, we observe that naive approach yields an mean error of 1.81 with a median of 1.52 when using all connections, and a slightly higher error for a MKSCS.

In Figure 4 we compare how different values for k influence the t-error and the planning time speedup when using a MKSCS. Generally, the t-error decreases as k increases, though we observe large differences between user types. For the cautious user the error does not exceed 1.1 on average, independent of k. While the aggressive user type achieves comparable values for $k = 100$, the error increases drastically for smaller k, reaching an average of ≈ 2.2 for $k = 25$ with a large deviation from the mean. The moderate user shows a more balanced result with the average not surpassing 1.5 for small k. Yet, the planning time speedup is less affected by the user type. For $k = 25$ we observe the highest speedup of ≈ 35 on the median for all three user types. For higher values of k the speedup decreases, but even using a connection set with $|\mathcal{E}| = 100$, the median path planning is at least 10 times faster than when using all connections $\mathcal{B}$. We conclude that using a MKSCS drastically decreases planning time though the cost increases

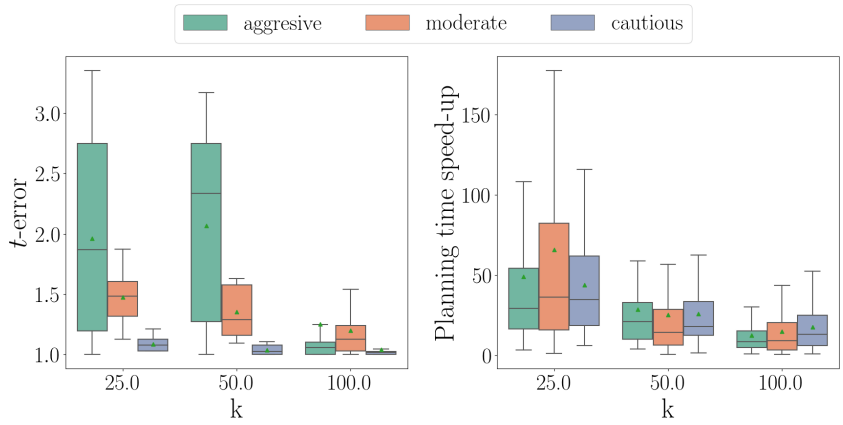


Fig. 4: Training data: The t-error and planning time speedup for different sizes of the MKSCS and the different user types.

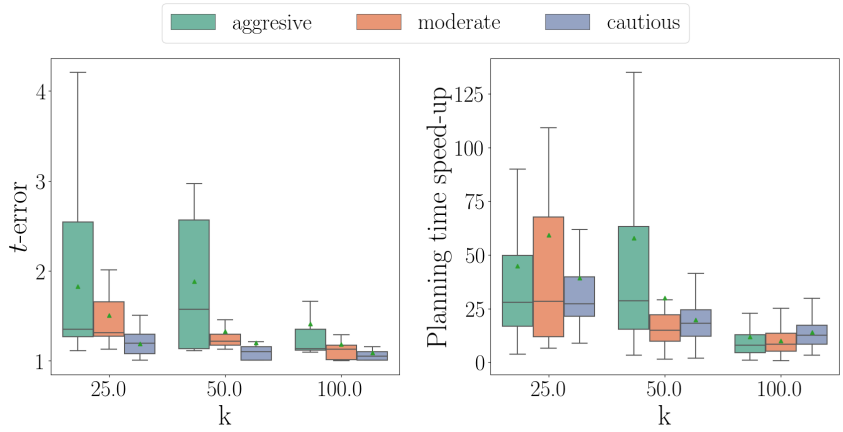


Fig. 5: Test data: The t-error and planning time speedup for different sizes of the MKSCS and the different user types.

34% compared to the optimum (twice the error of the best estimate). However, in half of all training examples the increase in cost is less than 10%. Between users, the performance benefit is similar, while the path quality differs.

4.2 Test Error

The test error is evaluated on three different start–goal pairs for which no demonstrations were obtained in two different environments. In Figure 3b we show a similar analysis as we did for the training error. The error of the learned control set $(\hat{\mathcal{E}}, \hat{\boldsymbol{w}})$ is slightly higher than in the training with a mean of 1.38 and a median of 1.17 while the deviation is smaller. The naive approach performs slightly better than in the training, with mean and median values for $(\mathcal{B}_{\texttt{naive}}, \boldsymbol{w}_{\texttt{naive}})$ of 1.72 and 1.31, respectively. This indicates that the user behaviour might be less

sensitive in the test scenarios. Figure 5 shows a comparable relationship between k and the t-error (left) and planning time speedup (right). The overall t-error is higher than in the training, and also increases with smaller k for the cautious user type. The aggressive user shows the highest error, with a mean of 1.8 for $k = 50$. The planning speedup shows a similar trend as the training data. However, the deviations differ more between user types. All three users yield median speedup factors of ≈ 30 for $k = 25$, but are still above 10 for $k = 100$.

We conclude that the learned control set $(\hat{\mathcal{E}}, \hat{\boldsymbol{w}})$ achieves good t-errors in both training and test environments while allowing for a substantial planning time speedup. Increasing k, the t-error tends to decrease, approaching the error of the estimation. Even though the performance also decreases with growing k, we still obtain an improvement of more than a factor of 10 for $k = 100$ compared to paths planned using the complete connection set $\mathcal{B}$.

5 Discussion

We studied a novel approach for computing a control set that captures user preferences. First an estimate $\hat{\boldsymbol{w}}$ of the user weights, given data, and a set of trajectories using those weights are computed for all pairs $(s, j) \in \mathcal{B}$. These trajectories are calculated to minimize a user cost function that is a linear combination of weights $\hat{\boldsymbol{w}}$ and trajectory features. Next, a set of connections $\hat{\mathcal{E}} \subseteq \mathcal{B}$ and its associated trajectories is determined. This control set minimizes the relative error of trajectories generated. We illustrate how both $\hat{\boldsymbol{w}}$ and $\hat{\mathcal{E}}$ are computed, and validate our findings using a clothoid trajectory planner. The results illustrate that the control set $(\hat{\mathcal{E}}, \hat{\boldsymbol{w}})$ is able to capture a user's preferences while greatly reducing online computation time relative to using the full connection set.

In future, this approach should be tested with real-world data, e.g., recorded driving behaviour. The use of other trajectory planners such as polynomial splines or Bezier curves should be used to investigate generalizability. Finally, these results should be extended to account for situational user weights.

References

1. M. Pivtoraiko and A. Kelly, "Kinodynamic motion planning with state lattice motion primitives," in *2011 IEEE/RSJ IROS*. IEEE, 2011, pp. 2172–2179.
2. M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.
3. M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *IJRR*, vol. 28, no. 8, pp. 933–945, 2009.
4. M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4889–4895.
5. C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

6. L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *IJRR*, vol. 34, no. 7, pp. 883–921, 2015.
7. T. Fraichard and A. Scheuer, "From reeds and shepp's to continuous-curvature paths," *IEEE Transactions on Robotics*, vol. 20, no. 6, pp. 1025–1035, 2004.
8. A. Botros and S. L. Smith, "Computing a minimal set of t-spanning motion primitives for lattice planners," in *2019 IEEE/RSJ IROS*, Nov 2019, pp. 2328–2335.
9. D. S. González, O. Erkent, V. Romero-Cano, J. Dibangoye, and C. Laugier, "Modeling driver behavior from demonstrations in dynamic environments using spatiotemporal lattices," in *2018 IEEE ICRA*. IEEE, 2018, pp. 1–7.
10. T. Gu, J. Atwood, C. Dong, J. M. Dolan, and J.-W. Lee, "Tunable and stable real-time trajectory planning for urban autonomous driving," in *2015 IEEE/RSJ IROS*. IEEE, 2015, pp. 250–256.
11. D. Sadigh, A. D. Dragan, S. Sastry, and S. A. Seshia, "Active preference-based learning of reward functions," in *Robotics: Science and Systems (RSS)*, 2017.
12. P. Abbeel, D. Dolgov, A. Y. Ng, and S. Thrun, "Apprenticeship learning for motion planning with application to parking lot navigation," in *2008 IEEE/RSJ IROS*. IEEE, 2008, pp. 1083–1090.
13. R. De Iaco, S. L. Smith, and K. Czarnecki, "Learning a lattice planner control set for autonomous vehicles," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 549–556.
14. M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 4889–4895.
15. P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 1.
16. M. Palan, N. C. Landolfi, G. Shevchuk, and D. Sadigh, "Learning reward functions by integrating human demonstrations and preferences," *arXiv preprint arXiv:1906.08928*, 2019.
17. E. Cheung, A. Bera, E. Kubin, K. Gray, and D. Manocha, "Identifying driver behaviors using trajectory features for vehicle navigation," in *2018 IEEE/RSJ IROS*. IEEE, 2018, pp. 3445–3452.
18. M. Rufli and R. Siegwart, "On the design of deformable input-/state-lattice graphs," in *2010 IEEE ICRA*. IEEE, 2010, pp. 3071–3077.
19. A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, vol. 42, no. 3, pp. 529–551, 2018.
20. B. J. Cohen, S. Chitta, and M. Likhachev, "Search-based planning for manipulation with motion primitives," in *2010 IEEE ICRA*. IEEE, 2010, pp. 2902–2908.
21. B. Korte and J. Vygen, *Combinatorial optimization*, 6th ed. Springer, 2018.
22. B. Michini, T. J. Walsh, A.-A. Agha-Mohammadi, and J. P. How, "Bayesian nonparametric reward learning from demonstration," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 369–386, 2015.
23. A. Kasperski and P. Zielinski, "An approximation algorithm for interval data minmax regret combinatorial optimization problems." *Inf. Process. Lett.*, vol. 97, no. 5, pp. 177–180, 2006.
24. L. Wasserman, *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013.