Optimal Decision-Making for Autonomous Agents via Data Composition

Émiland Garrabé, Martina Lamberti and Giovanni Russo*

Abstract

We consider the problem of designing agents able to compute optimal decisions by composing data from multiple sources to tackle tasks involving: (i) tracking a desired behavior while minimizing an agent-specific cost; (ii) satisfying safety constraints. After formulating the control problem, we show that this is convex under a suitable assumption and find the optimal solution. The effectiveness of the results, which are turned in an algorithm, is illustrated on a *connected cars* application via in-silico and in-vivo experiments with real vehicles and drivers. All the experiments confirm our theoretical predictions and the deployment of the algorithm on a real vehicle shows its suitability for in-car operation.

I. INTRODUCTION

We often make decisions by composing knowledge gathered from *others* [1] and a challenge transversal to control and learning is to devise mechanisms allowing autonomous decision-makers to emulate these abilities. Systems based on sharing data [2] are examples where agents need to make decisions based on some form of crowdsourcing [3]. Similar mechanisms can also be useful for the data-driven control paradigm when e.g., one needs to re-use policies synthesized on plants for which data are available to solve a control task on a new plant, for which data are scarcely available [3]–[5].

Motivated by this, we consider the problem of designing decision-making mechanisms that enable autonomous agents to compute optimal decisions by composing information from third parties to solve tasks that involve: (i) tracking a desired behavior while minimizing an agent-specific cost; (ii) satisfying safety constraints. Our results enable computation of the optimal behavior and are turned into an algorithm. This is experimentally validated on a *connected car* application.

Related works: we briefly survey a number of works related to the results and methodological framework of this paper. The design of context-aware switches between multiple datasets for autonomous agents has been recently considered in [3], [4], where the design problem, formalized as a data-driven control (DDC) problem, did not take into account safety requirements. Results in DDC include [6]–[8], which take a behavioral systems perspective, [9], which finds data-driven formulas towards a model-free theory of geometric control. We also recall e.g., [10]–[12] for results inspired from MPC, [5] that considers data-driven control policies transfer and [13] that tackles he problem of computing data-driven minimum-energy control for linear systems. In our control problem (see Section III) we formalize the tracking of a given behavior via Kullback-Leibler (KL) divergence minimization and we refer to e.g., [14], [15] for examples across learning and control that involve minimizing this functional. Further, the study of mechanisms enabling agents to re-use data, also arises in the design of prediction algorithms from experts [16] and of learning algorithms from multiple simulators [17]. In a yet broader context, studies in neuroscience [18] hint that our neocortex might implement a mechanism composing the output of the cortical columns and this might be the basis of our ability to re-use knowledge.

Contributions: we consider the problem of designing agents that dynamically combine data from heterogeneous sources to fulfill tasks that involve tracking a target behavior while optimizing a cost and satisfying safety requirements expressed as box constraints. By leveraging a probabilistic framework, we formulate a data-driven optimal control problem and, for this problem, we: (i) prove convexity under a suitable condition; (ii) find the optimal solution; (iii) turn our results into an algorithm, using it to design an intelligent parking system for connected vehicles. Validations are performed both *in-silico* and *in-vivo*, with real cars. As such, the main purpose of this paper is twofold: (i) we introduce, and rigorously characterize our algorithm; (ii) propose a stand-alone implementation of our results, suitable for in-vivo experiments on real cars. In-vivo validations were performed via an hardware-in-the-loop platform allowing to embed real cars/drivers in the experiments. Using the platform, we deploy our algorithm on a real vehicle showing its suitability for in-car operation. All experiments confirm the effectiveness of our approach (documented code/data for our simulations at `https://tinyurl.com/3ep4pknh`).

While our results are inspired by the ones in [3], [4], our paper extends these in several ways. First, the results in [3], [4] cannot consider box constraints and hence cannot tackle the control problem of this paper. Second, even when there are no box constraints, the results in [3], [4] only solve an approximate version of the problem considered here. That is, the results from [3], [4] only find an approximate, non-optimal, solution of the problem considered here. As we shall see, this means that the solutions from [3], [4] cannot get a better cost than the one obtained with the results of this paper. Third, the algorithm obtained from the results in this paper is deployed, and validated, on a real car and this was not done in [3], [4]. The *in-vivo* implementation is novel.

* emails: {`egarrabe,giovarusso`}`@unisa.it`, `martinalamberti3@gmail.com`. E. Garrabé and G. Russo with the DIEM at the University of Salerno, 84084, Salerno, Italy.

Notation: sets are in *calligraphic* and vectors in **bold**. Given the measurable space $(\mathcal{X}, \mathcal{F}_x)$, with $\mathcal{X} \subseteq \mathbb{R}^d$ ($\mathcal{X} \subseteq \mathbb{Z}^d$) and $\mathcal{F}_x$ being a σ-algebra on $\mathcal{X}$, a random variable on $(\mathcal{X}, \mathcal{F}_x)$ is denoted by $\mathbf{X}$ and its realization by $\mathbf{x}$. The *probability density (resp. mass) function* or *pdf* (*pmf*) of a continuous (discrete) $\mathbf{X}$ is denoted by $p(\mathbf{x})$. The convex subset of such probability functions (pfs) is $\mathcal{D}$. The expectation of a function $\mathbf{h}(\cdot)$ of the continuous variable $\mathbf{X}$ is $\mathbb{E}_p[\mathbf{h}(\mathbf{X})] := \int \mathbf{h}(\mathbf{x})p(\mathbf{x})d\mathbf{x}$, where the integral (in the sense of Lebesgue) is over the support of $p(\mathbf{x})$, which we denote by $\mathcal{S}(p)$. The joint pf of $\mathbf{X}_1$, $\mathbf{X}_2$ is $p(\mathbf{x}_1, \mathbf{x}_2)$ and the conditional pf of $\mathbf{X}_1$ given $\mathbf{X}_2$ is $p(\mathbf{x}_1 \mid \mathbf{x}_2)$. Countable sets are denoted by $\{w_k\}_{k_1:k_n}$, where w_k is the generic element of the set and $k_1 : k_n$ is the closed set of consecutive integers between k_1 and k_n. The KL divergence between $p(\mathbf{x})$ and $q(\mathbf{x})$, where p is absolutely continuous w.r.t. q, is $\mathcal{D}_{\mathrm{KL}}(p \mid\mid q) := \int_{\mathcal{S}(p)} p \, \ln(p/q) \, d\mathbf{x}$: it is non-negative and 0 if and only if $p(\mathbf{x}) = q(\mathbf{x})$. In the expressions for the expectation and KL divergence, the integral is replaced by the sum if the variables are discrete. Finally: (i) we let $\mathbb{1}_{\mathcal{A}}(\mathbf{x})$ denote the indicator function being equal to 1 if $\mathbf{x} \in \mathcal{A} \subseteq \mathcal{X}$ and 0 otherwise; (ii) set exclusion is instead denoted by $\backslash$.

II. THE SETUP

The agent seeks to craft its behavior by combining a number of sources to fulfill a task that involves tracking a target/desired behavior while maximizing an agent-specific reward over the time horizon $\mathcal{T} := 0 : T$, $T > 0$. The agent's state at time step $k \in \mathcal{T}$ is $\mathbf{x}_k \in \mathcal{X}$ and the target behavior that the agent seeks to track is $p_{0:T} := p_0(\mathbf{x}_0) \prod_{k \in 1:T} p(\mathbf{x}_k \mid \mathbf{x}_{k-1})$. As in [3], [4], we design the behavior of the agent by designing its joint pf $\pi_{0:T} := \pi(\mathbf{x}_0, \ldots, \mathbf{x}_T)$ and we have:

$$\pi_{0:T} = \pi_0(\mathbf{x}_0) \prod_{k \in 1:T} \pi(\mathbf{x}_k \mid \mathbf{x}_{k-1}). \tag{1}$$

That is, the behavior of the agent can be designed via the pfs $\pi(\mathbf{x}_k \mid \mathbf{x}_{k-1})$, i.e., the transition probabilities. To do so, the agent has access to S sources and we denote by $\pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})$, with support $\mathcal{S}(\pi) \subseteq \mathcal{X}$, the behavior made available by source i, $i \in \mathcal{S} := 1 : S$, at $k - 1$. We also let $r_k(\mathbf{x}_k)$ be the agent's reward for being in state $\mathbf{x}_k$ at k.

III. FORMULATION OF THE CONTROL PROBLEM

Let $\alpha_k^{(1)}, \ldots, \alpha_k^{(S)}$ be weights and $\boldsymbol{\alpha}_k$ be their stack. Then, the control problem we consider can be formalized as:
Problem 1: find the sequence $\{\boldsymbol{\alpha}_k^*\}_{1:T}$ solving

$$\min_{\{\boldsymbol{\alpha}_k\}_{1:T}} \mathcal{D}_{\mathrm{KL}}(\pi_{0:T} \mid\mid p_{0:T}) - \sum_{k=1}^{T} \mathbb{E}_{\pi(\mathbf{x}_{k-1})}[\tilde{r}_k(\mathbf{X}_{k-1})]$$

$$\text{s.t. } \mathbb{E}_{\pi(\mathbf{x}_k \mid \mathbf{x}_{k-1})}[\mathbb{1}_{\mathcal{X}_k}(\mathbf{x}_k)] \geq 1 - \epsilon_k, \quad \forall k,$$

$$\pi(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}), \quad \forall k,$$

$$\sum_{i \in \mathcal{S}} \alpha_k^{(i)} = 1, \quad \alpha_k^{(i)} \in [0, 1], \quad \forall k.$$

In Problem 1, $\tilde{r}_k(\mathbf{x}_{k-1}) := \mathbb{E}_{\pi(\mathbf{x}_k \mid \mathbf{x}_{k-1})}[r_k(\mathbf{X}_k)]$ and we note that $\mathbb{E}_{\pi(\mathbf{x}_{k-1})}[\tilde{r}_k(\mathbf{X}_{k-1})] = \mathbb{E}_{\pi(\mathbf{x}_k)}[r_k(\mathbf{X}_k)]$ is the expected reward for the agent when the behavior in (1) is followed. The problem is a finite-horizon optimal control problem with the $\boldsymbol{\alpha}_k$'s as decision variables. As we shall see, these are generated as feedback from the agent state (Section IV). We say that $\{\pi^*(\mathbf{x}_k \mid \mathbf{x}_{k-1})\}_{1:T}$, with $\pi^*(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \sum_{i \in \mathcal{S}} \alpha_k^{(i),*} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})$, is the optimal behavior for the agent, obtained by linearly combining the sources via the $\boldsymbol{\alpha}_k^*$'s. In the problem, the cost formalizes the fact that the agent seeks to maximize its reward, while tracking (in the KL divergence sense) the target behavior. Minimizing the KL term amounts at minimizing the discrepancy between $\pi_{0:T}$ and $p_{0:T}$. This term can also be thought as a divergence regularizer and, when $p_{0:T}$ is uniform, it becomes an entropic regularizer. The second and third constraints formalize the fact that, at each k, $\pi^*(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \in \mathcal{D}$ and that this is a convex combination of the pfs from the sources. The first constraint is a box constraint and models the fact that the probability that the agent behavior is, at each k, inside some (e.g., safety) measurable set $\mathcal{X}_k \subseteq \mathcal{X}$ is greater than some $\epsilon_k \geq 0$. We now make the following

Assumption 1: the optimal cost of Problem 1 is bounded.

Remark 1: the cost in Problem 1 can be recast as $\mathcal{D}_{\mathrm{KL}}(\pi_{0:T} \mid\mid \tilde{p}_{0:T})$, where $\tilde{p}_{0:T} \propto p_{0:T} \exp\left(\sum_{k=1}^{T} r_k(\mathbf{x}_k)\right)$. This means that Assumption 1 is satisfied whenever there exists some $\tilde{\pi}_{0:T}$ that is feasible for Problem 1 and that is absolutely continuous w.r.t. $\tilde{p}_{0:T}$. See also Remark 3.

IV. MAIN RESULTS

We propose an algorithm to tackle Problem 1. The algorithm takes as input T, the target behavior, the reward, the behaviors of the sources and the box constraints of Problem 1 (if any). Given the inputs, it returns the optimal behavior for the agent.

The key steps of the algorithm are given as pseudo-code in Algorithm 1. An agent that follows Algorithm 1 computes $\{\boldsymbol{\alpha}_k\}_{1:N}$ via backward recursion (lines $4-9$). At each k, the $\boldsymbol{\alpha}_k$'s are obtained as the minimizers of

$$
\begin{aligned}
\min_{\boldsymbol{\alpha}_k} \; & c_k(\pi(\mathbf{x}_k \mid \mathbf{x}_{k-1})) \\
\text{s.t. } & \mathbb{E}_{\pi(\mathbf{x}_k \mid \mathbf{x}_{k-1})}\left[\mathbb{1}_{\mathcal{X}_k}(\mathbf{x}_k)\right] \geq 1 - \epsilon_k \\
& \pi(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \\
& \sum_{i \in \mathcal{S}} \alpha_k^{(i)} = 1, \quad \alpha_k^{(i)} \in [0,1],
\end{aligned}
\tag{2}
$$

where

$$
\begin{aligned}
c_k(\pi(\mathbf{x}_k \mid \mathbf{x}_{k-1})) := \; & \mathcal{D}_{\mathrm{KL}}\left(\pi(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \,\|\, p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\right) \\
& - \mathbb{E}_{\pi(\mathbf{x}_k \mid \mathbf{x}_{k-1})}\left[\bar{r}_k(\mathbf{X}_k)\right],
\end{aligned}
\tag{3}
$$

with $\bar{r}_k(\cdot)$ iteratively built within the recursion (lines 5, 8). The weights are used (line 7) to compute $\pi^*(\mathbf{x}_k \mid \mathbf{x}_{k-1})$.

Remark 2: results are stated for continuous variables (proofs for discrete variables omitted for brevity). Note that integrals/summations in the cost are over $\mathcal{S}(\pi)$.

Remark 3: following Remark 1, the optimal cost of the problem in (2) is bounded if there exists some feasible $\tilde{\pi}(\mathbf{x}_k \mid \mathbf{x}_{k-1})$ that is absolutely continuous w.r.t. $\tilde{p}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \propto p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \exp\left(\bar{r}_k(\mathbf{x}_k)\right)$. From the design viewpoint, this can satisfied if it holds for at least one $\pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})$.

Finally, we make the following

Assumption 2: $\forall i \in \mathcal{S}$ and $\forall \mathbf{x}_{k-1} \in \mathcal{X}$, there exist some constants, say m and M, with $0 < m \leq M < +\infty$, such that $m \leq \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \leq M$, $\forall \mathbf{x}_k \in \mathcal{S}(\pi)$.

Remark 4: Assumption 1 is satisfied for e.g., Gaussian distributions. As we shall see (Section V) the assumption can be fulfilled by injecting noise in the data.

A. Properties of Algorithm 1

Before characterizing convexity of the problems recursively solved in Algorithm 1 and optimality, we give a condition ensuring feasibility of the problem in (2).

Lemma 1: the problem in (2) is feasible if and only if there exists at least one source, say $\pi^{(j)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})$, such that $\mathbb{E}_{\pi^{(j)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})}\left[\mathbb{1}_{\mathcal{X}_k}(\mathbf{x}_k)\right] \geq 1 - \epsilon_k$.

Proof: the *if* part clearly holds. For the *only if* part we prove that if problem (2) is infeasible, then $\max_i \mathbb{E}_{\pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})}\left[\mathbb{1}_{\mathcal{X}_k}(\mathbf{x}_k)\right] < 1 - \epsilon_k$. In fact, if the problem is infeasible, then for all $\boldsymbol{\alpha}_k$ such that $\sum_{i \in \mathcal{S}} \alpha_k^{(i)} = 1$ and $\alpha_k^{(i)} \in [0,1]$ it must hold that

$$
\int_{\mathcal{X}_k} \sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) d\mathbf{x}_k < 1 - \epsilon_k.
$$

Note that this must also hold for all $\boldsymbol{\alpha}_k$ such that $\sum_{i \in \mathcal{S}} \alpha_k^{(i)} = 1$ and $\alpha_k^{(i)} \in \{0,1\}$, as these are contained in the set of real-valued $\boldsymbol{\alpha}_k$'s. We conclude the proof by noticing that, if $\boldsymbol{\alpha}_k$ is such that $\alpha_k^{(i)} = 0 \forall i \neq j$ and $\alpha_k^{(j)} = 1$, then

$$
\begin{aligned}
\int_{\mathcal{X}_k} \sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) d\mathbf{x}_k &= \int_{\mathcal{X}_k} \pi^{(j)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) d\mathbf{x}_k \\
&= E_{\pi^{(j)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})}\left[\mathbb{1}_{\mathcal{X}_k}(\mathbf{x}_k)\right].
\end{aligned}
$$

Algorithm 1 Pseudo-code

1: **Input:** time horizon T, target behavior $p_{0:T}$, reward $r_k(\cdot)$, sources $\pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})$, box constraints (optional)
2: **Output:** optimal agent behavior $\{\pi^*(\mathbf{x}_k \mid \mathbf{x}_{k-1})\}_{1:T}$
3: $\hat{r}_T(\mathbf{x}_T) \leftarrow 0$
4: **for** $k = T : 1$ **do**
5: $\quad \bar{r}_k(\mathbf{x}_k) \leftarrow r_k(\mathbf{x}_k) - \hat{r}_k(\mathbf{x}_k)$
6: $\quad \boldsymbol{\alpha}_k^*(\mathbf{x}_{k-1}) \leftarrow$ minimizer of the problem in (2);
7: $\quad \pi^*(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \leftarrow \sum_{i \in \mathcal{S}} \alpha_k^{(i),*}(\mathbf{x}_{k-1}) \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})$
8: $\quad \hat{r}_{k-1}(\mathbf{x}_{k-1}) \leftarrow c_k(\pi^*(\mathbf{x}_k \mid \mathbf{x}_{k-1}))$
9: **end for**

It then follows that, $\forall j$,

$$E_{\pi^{(j)}(\mathbf{x}_k | \mathbf{x}_{k-1})} \left[\mathbb{1}_{\mathcal{X}_k}(\mathbf{x}_k) \right] < 1 - \epsilon_k.$$

■

Convexity: we are now ready to prove the following

Proposition 1: let Assumption 2 hold. Then, the problem in (2) is convex.

Proof: Clearly, the second and third constraint in the problem are convex. For the first constraint, we get

$$\mathbb{E}_{\pi(\mathbf{x}_k | \mathbf{x}_{k-1})} \left[\mathbb{1}_{\mathcal{X}_k}(\mathbf{x}_k) \right] = \int \pi(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \mathbb{1}_{\mathcal{X}_k}(\mathbf{x}_k) d\mathbf{x}_k$$

$$= \int_{\mathcal{X}_k} \sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) d\mathbf{x}_k$$

$$= \sum_{i \in \mathcal{S}} \alpha_k^{(i)} \int_{\mathcal{X}_k} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) d\mathbf{x}_k,$$

which is therefore convex in the decision variables.

Now, we show that the cost is also convex in these variables and we do so by explicitly computing, for each $\mathbf{x}_{k-1}$, its Hessian, say $\mathbf{H}(\mathbf{x}_{k-1})$. Specifically, after embedding the second constraint of the problem in (2) in the cost and differentiating with respect to the decision variables we get, for each $j \in \mathcal{S}$:

$$\frac{\partial c_k}{\partial \alpha_k^{(j)}} := \frac{\partial}{\partial \alpha_k^{(j)}} \int_{\mathcal{S}(\pi)} \sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \left(\log \left(\frac{\sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})}{p(\mathbf{x}_k \mid \mathbf{x}_{k-1})} \right) - \bar{r}_k(\mathbf{x}_k) \right) d\mathbf{x}_k$$

$$= \int_{\mathcal{S}(\pi)} \frac{\partial}{\partial \alpha_k^{(j)}} \sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \left(\log \left(\frac{\sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})}{p(\mathbf{x}_k \mid \mathbf{x}_{k-1})} \right) - \bar{r}_k(\mathbf{x}_k) \right) d\mathbf{x}_k$$

$$= \int_{\mathcal{S}(\pi)} \pi^{(j)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \left(\log \left(\sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \right) - \log \left(p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \right) - \bar{r}_k(\mathbf{x}_k) + 1 \right) d\mathbf{x}_k.$$

The above chain of identities was obtained by swapping integration and differentiation, leveraging the fact that the cost is smooth in the decision variables. Similarly, we get

$$\frac{\partial^2 c_k}{\partial \alpha_k^{(j)2}} = \frac{\partial}{\partial \alpha_k^{(j)}} \int_{\mathcal{S}(\pi)} \pi^{(j)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \left(\log \left(\sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \right) - \log \left(p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \right) - \bar{r}_k(\mathbf{x}_k) + 1 \right) d\mathbf{x}_k$$

$$= \int_{\mathcal{S}(\pi)} \frac{\pi^{(j)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})^2}{\sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})} d\mathbf{x}_k$$

$$=: h_{jj}(\mathbf{x}_{k-1}),$$

and, for each $m \neq j$, $m \in \mathcal{S}$,

$$\frac{\partial^2 c_k}{\partial \alpha_k^{(j)} \partial \alpha_k^{(m)}} = \int_{\mathcal{S}(\pi)} \frac{\pi^{(j)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \pi^{(m)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})}{\sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})} d\mathbf{x}_k =: h_{jm}(\mathbf{x}_{k-1}).$$

Also, following Assumption 2, $\forall j, m \in \mathcal{S}$ we have that

$$\int_{\mathcal{S}(\pi)} \left| \frac{\pi^{(j)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \pi^{(m)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})}{\sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})} \right| d\mathbf{x}_k \leq \frac{M}{m}$$

where we used the third constraint. That is, the above integrals are well defined and thus we can conclude the proof by computing $\mathbf{v}^T \mathbf{H}(\mathbf{x}_{k-1}) \mathbf{v}$ for some non-zero $\mathbf{v} \in \mathbb{R}^S$:

$$\mathbf{v}^T \mathbf{H}(\mathbf{x}_{k-1}) \mathbf{v} = \sum_{j,m} v_j v_m h_{jm}(\mathbf{x}_{k-1})$$

$$= \sum_{j,m} v_j v_m \int_{\mathcal{S}(\pi)} a_{jm}(\mathbf{x}_k, \mathbf{x}_{k-1}) d\mathbf{x}_k$$

$$= \int_{\mathcal{S}(\pi)} \sum_{j,m} v_j v_m a_{jm}(\mathbf{x}_k, \mathbf{x}_{k-1}) d\mathbf{x}_k,$$

where the a_{jm}'s are the elements of the matrix

$$A(\mathbf{x}_k, \mathbf{x}_{k-1}) := \bar{\pi}(\mathbf{x}_k, \mathbf{x}_{k-1}) \begin{bmatrix} \pi^{(1)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \\ \vdots \\ \pi^{(S)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \end{bmatrix} \cdot \cdot \begin{bmatrix} \pi^{(1)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \dots \pi^{(S)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \end{bmatrix},$$

with $\bar{\pi}(\mathbf{x}_k, \mathbf{x}_{k-1}) := 1/\left(\sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})\right)$. The above expression is indeed positive semi-definite for each $\mathbf{x}_k$, $\mathbf{x}_{k-1}$ and we can draw the desired conclusion. ∎

Optimality: we can now prove the following

Proposition 2: let Assumption 2 and Assumption 1 hold. Then, Algorithm 1 gives an optimal solution for Problem 1.

Proof: the chain rule for the KL divergence and the linearity of expectation imply that the cost can be written as

$$\mathcal{D}_{\mathrm{KL}}\left(\pi_{0:T-1} \mid\mid p_{0:T-1}\right) - \sum_{k=1}^{T-1} \mathbb{E}_{\pi(\mathbf{x}_{k-1})}\left[\tilde{r}_k(\mathbf{X}_{k-1})\right] + \mathbb{E}_{\pi(\mathbf{x}_{T-1})}\left[c_T(\pi(\mathbf{x}_T \mid \mathbf{x}_{T-1}))\right], \tag{4}$$

where $c_T(\pi(\mathbf{x}_T \mid \mathbf{x}_{T-1}))$ is defined as in (3) with $\bar{r}_T(\mathbf{x}_T)$ given by Algorithm 1 – see lines 3 and 5 and note that, at time step T, $\bar{r}_T(\mathbf{x}_T) = r_T(\mathbf{x}_T)$. To obtain the above expression, the fact that $c_T(\pi(\mathbf{x}_T \mid \mathbf{x}_{T-1}))$ only depends on $\mathbf{x}_{T-1}$ was also used. Hence, Problem 1 can be split into the sum of two sub-problems: a first problem over $k \in 0 : T - 1$ and the second for $k = T$. For this last time step, the problem can be solved independently on the others and is given by:

$$\min_{\boldsymbol{\alpha}_T} \; \mathbb{E}_{\pi(\mathbf{x}_{T-1})}[c_T(\pi(\mathbf{x}_T \mid \mathbf{x}_{T-1}))]$$
$$\text{s.t. } \mathbb{E}_{\pi(\mathbf{x}_T \mid \mathbf{x}_{T-1})}\left[\mathbb{1}_{\mathcal{X}_T}(\mathbf{x}_T)\right] \geq 1 - \epsilon_T,$$
$$\pi(\mathbf{x}_T \mid \mathbf{x}_{T-1}) = \sum_{i \in \mathcal{S}} \alpha_T^{(i)} \pi^{(i)}(\mathbf{x}_T \mid \mathbf{x}_{T-1}), \tag{5}$$
$$\sum_{i \in \mathcal{S}} \alpha_T^{(i)} = 1, \;\; \alpha_T^{(i)} \in [0,1].$$

Using linearity of the expectation and the fact that the decision variable is independent on $\pi(\mathbf{x}_{T-1})$, we have that the minimizer of the problem in (5) is the same as the problem in (2) with $k = T$. Following Proposition 1, such a problem is convex and we denote its optimal solution as $\boldsymbol{\alpha}_T^*(\mathbf{x}_{T-1})$ – see line 6 of Algorithm 1 – and the optimal cost of the problem, which is bounded by Assumption 1, is $c_T(\pi^*(\mathbf{x}_T \mid \mathbf{x}_{T-1}))$, where:

$$\pi^*(\mathbf{x}_T \mid \mathbf{x}_{T-1}) = \sum_{i \in \mathcal{S}} \boldsymbol{\alpha}_T^{(i),*}(\mathbf{x}_{T-1}) \pi^{(i)}(\mathbf{x}_T \mid \mathbf{x}_{T-1}).$$

This gives $\hat{r}_{T-1}(\mathbf{x}_{T-1})$ in Algorithm 1 (lines $7-8$), thus yielding the steps for the backward recursion of the Algorithm 1 at time step T. Now, the minimum value of the problem in (5) is given by $\mathbb{E}_{\pi(\mathbf{x}_{T-1})}\left[\hat{r}_{T-1}(\mathbf{X}_{T-1})\right]$. Hence, the cost of Problem 1 becomes

$$\mathcal{D}_{\mathrm{KL}}\left(\pi_{0:T-1} \mid\mid p_{0:T-1}\right) - \sum_{k=1}^{T-1} \mathbb{E}_{\pi(\mathbf{x}_{k-1})}\left[\tilde{r}_k(\mathbf{X}_{k-1})\right] + \mathbb{E}_{\pi(\mathbf{x}_{T-1})}\left[\hat{r}_{T-1}(\mathbf{X}_{T-1})\right].$$

Then, following the same reasoning used to obtain (4) and by noticing that

$$\mathbb{E}_{\pi(\mathbf{x}_{T-1})}\left[\hat{r}_{T-1}(\mathbf{X}_{T-1})\right] = \mathbb{E}_{\pi(\mathbf{x}_{T-2})}\left[\mathbb{E}_{\pi(\mathbf{x}_{T-1} \mid \mathbf{x}_{T-2})}\left[\hat{r}_{T-1}(\mathbf{X}_{T-1})\right]\right],$$

we get:

$$\mathcal{D}_{\mathrm{KL}}\left(\pi_{0:T-2} \mid\mid p_{0:T-2}\right) - \sum_{k=1}^{T-2} \mathbb{E}_{\pi(\mathbf{x}_{k-1})}\left[\tilde{r}_k(\mathbf{X}_{k-1})\right] + \mathbb{E}_{\pi(\mathbf{x}_{T-2})}\left[c_{T-1}(\pi(\mathbf{x}_{T-1} \mid \mathbf{x}_{T-2}))\right],$$

where $c_{T-1}(\pi(\mathbf{x}_{T-1} \mid \mathbf{x}_{T-2}))$ is again given in (3) with $\bar{r}_{T-1}(\mathbf{x}_{T-1})$ again defined as in Algorithm 1. By iterating the arguments above, we find that at each time step Problem 1 can always be split as the sum of two sub-problems, where the last sub-problem can be solved independently on the previous ones. Moreover, the minimizer of this last sub-problem is

always the solution of a problem of the form

$$\min_{\boldsymbol{\alpha}_k} \mathcal{D}_{\text{KL}}\left(\pi(\mathbf{x}_k \mid \mathbf{x}_{k-1}) \| p(\mathbf{x}_k \mid \mathbf{x}_{k-1})\right) - \mathbb{E}_{\pi(\mathbf{x}_k \mid \mathbf{x}_{k-1})}\left[\bar{r}_k(\mathbf{X}_k)\right]$$

$$\text{s.t. } \mathbb{E}_{\pi(\mathbf{x}_k \mid \mathbf{x}_{k-1})}\left[\mathbb{1}_{\mathcal{X}_k}(\mathbf{x}_k)\right] \geq 1 - \epsilon_k,$$

$$\pi(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \sum_{i \in \mathcal{S}} \alpha_k^{(i)} \pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1}), \tag{6}$$

$$\sum_{i \in \mathcal{S}} \alpha_k^{(i)} = 1, \quad \alpha_k^{(i)} \in [0, 1],$$

where $\bar{r}_k(\mathbf{x}_k) := r_k(\mathbf{x}_k) - \hat{r}_k(\mathbf{x}_k)$, with $\hat{r}_k(\mathbf{x}_k) := c_{k+1}(\pi^*(\mathbf{x}_{k+1} \mid \mathbf{x}_k))$ and $\pi^*(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \sum_{i \in \mathcal{S}} \boldsymbol{\alpha}_k^{(i),*}(\mathbf{x}_{k-1})\pi^{(i)}(\mathbf{x}_k \mid \mathbf{x}_{k-1})$. This yields the desired conclusions. ∎

Remark 5: the results from [3], [4] solve an approximate version of Problem 1 when there are no box constraints. Hence, even in this special case, the results from [3], [4] do not lead to the optimal solution found with Algorithm 1. Specifically, the approximate solution from [3], [4] corresponds to the optimal solution of a problem with additional binary constraints on the decision variables. As a result, the algorithm from [3], [4] searches solutions in a feasibility domain that is contained in the feasibility domain of Problem 1. Hence, the solutions found in [3], [4] cannot achieve a better cost than the ones obtained via Algorithm 1.

V. DESIGNING AN INTELLIGENT PARKING SYSTEM

We now use Algorithm 1 to design an intelligent parking system for connected cars and validate the results via in-silico and in-vivo experiments. For the latter set of experiments, Algorithm 1 is deployed on a real car and validation is performed via an hardware-in-the-loop (HiL) platform inspired from [19]. Before reporting the results, we describe the validation scenarios and the experimental set-up. Code, maps and parameters with instructions to replicate the simulations are given at `https://tinyurl.com/3ep4pknh`.

A. Validation Scenarios and Experimental Set-up

We consider the problem of routing vehicles in a given geographic area to find parking. In all experiments we consider a morning rush scenario at the University of Salerno campus (see Figure 1). Specifically, cars arrive to the campus through a highway exit and, from here, users seek to park in one of the parking locations: *Biblioteca* and *Terminal*.

In this context, vehicles are agents equipped with Algorithm 1. The set of road links within the campus is $\mathcal{X}$ and time steps are associated to the time instants when the vehicle changes road link. The state of the agent, x_k, is the road link occupied by the vehicle at time step k. Given this set-up, at each k Algorithm 1 outputs the turning probability for the car given the current car link, $\pi^*(x_k \mid x_{k-1})$. The next direction for the car is then obtained by sampling from this pf.
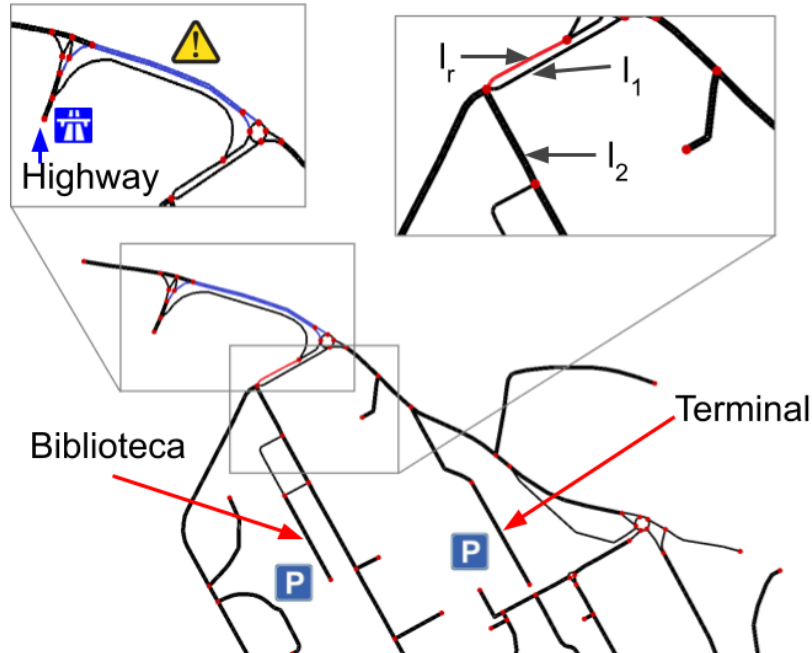

Fig. 1: campus map. The magnified areas show the obstructed road link (in blue) and links used within the validations. Colors online.

Agents have access to a set of sources, each providing different routes. As discussed in [20], sources might be third parties navigation services, routes collected from other cars/users participating to some sharing service. Agents wish to track their target/desired behavior (driving them to the preferred parking – Terminal in our experiments) and the reward depends on the actual road conditions within the campus. Links adjacent to a parking lot are assigned a constant reward of: (i) 3.8 if the parking has available spaces; (ii) 0 when it becomes full. Unparked cars already on full parking lots are assigned a target behavior leading them to another parking. In normal road conditions, the reward for the other links is 0 and becomes -20 when there is an obstruction. In our experiments, the reward was selected heuristically so that it would be: (i) sufficiently penalizing for links affected by obstruction; (ii) encouraging cars to drive towards parking lots with available spaces. In the first scenario (Scenario 1) there are no box constraints: this is done to benchmark Algorithm 1 with [3], [4]. To this aim, we use the campus map from [20] in which [3], [4] were thoroughly validated via simulations. Then, we show that by introducing box constraints Algorithm 1 can effectively regulate access of vehicles through selected road links. This is Scenario 2 and we considered three situations: (A) the road towards the *Biblioteca* parking lot is forbidden. To account for this, we set in Algorithm 1 $\mathcal{X}_k = \mathcal{X} \setminus l_2$, where the link l_2 is shown in Figure 1, and $\epsilon_k = 0.027$; (B) the set $\mathcal{X}_k$ as before but now $\epsilon_k = 0.5$; (C) the road towards the *Terminal* parking lot is forbidden and in this case we set $\mathcal{X}_k = \mathcal{X} \setminus l_1$, $\epsilon_k = 0.027$ (see Figure 1 for link l_1). For this last scenario, Algorithm 1 is validated both in-silico and in-vivo. Next, we describe the simulation and HiL experimental set-ups.

Simulation set-up: simulations were performed in SUMO [21]; see also [20] for a description of the pipeline to import maps and traffic demands. In our simulations, each parking lot can accommodate up to 50 cars and we generated the traffic demand so that 100 cars would arrive on campus at 5-second intervals. All the cars seek to park and, by doing so, the parking capacity is saturated. Given this setting, we simulated a road obstruction on the main link (in blue in Figure 1) from the highway exit to the campus entrance. This was done by restricting, in SUMO, the speed of the link to less than one kilometer per hour. Information on the cars in the simulation are contained in the stand-alone file `agent.npy`. Instead, the pfs associated with the sources (described below) are all stored in `behaviors.npy`.

HiL set-up: the platform embeds a real vehicle into a SUMO simulation. By doing so, performance of the algorithm deployed on a real car can be assessed under arbitrary synthetic traffic conditions generated via SUMO. The high-level architecture of the platform is shown in Figure 2. The platform creates an avatar of the real car in the SUMO simulation. Namely, as shown in Figure 2, the position of the real car is embedded in SUMO by using a standard smartphone to collect its GPS coordinates. These coordinates are then sent via bluetooth to a computer, also hosted on the car in the current implementation. The connection is established via an off-the-shelf app, which writes the coordinates in a `rfcomm` file. By using the `pySerial` library, an interface was developed to read data in Python. Here, a script was designed leveraging `pynmeaGPS` to translate the data in the `NMEA` format for longitude/latitude coordinates. With this data format, a Python script was created to place the avatar of the real car in the position given by the coordinates. A GUI is also included to highlight the trajectory of the real car on the map and an off-the-shelf text-to-speech routine is used to provide audio feedback to the driver on the vehicle.

B. In-car Implementation of the Algorithm

For both the in-silico and in-vivo experiments, Algorithm 1 was implemented in Python as a stand-alone class so that each car equipped with the algorithm could function as a stand-alone agent. The class has methods accepting all the input parameters of Algorithm 1 and providing as output the car behavior computed by the algorithm. The optimization problems within the algorithm loop were solved via the Python library `scipy.optimize`. Additionally, the class also implements methods to compute the cost and to support receding horizon implementations of Algorithm 1. In our experiments, we used this receding horizon implementation: the width of the receding horizon window was $T = 5$ and every time the car entered
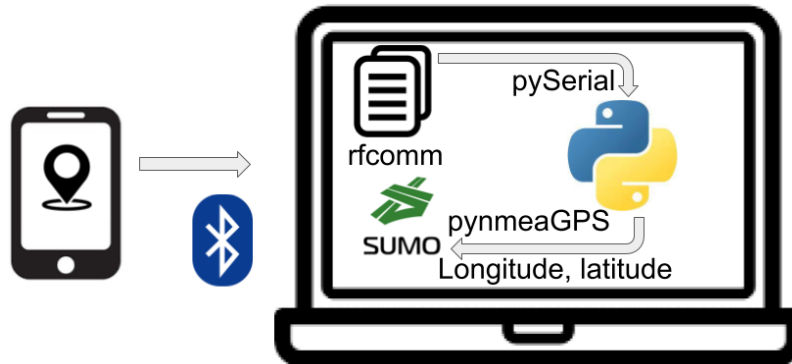

Fig. 2: HiL functional architecture.

in a new link/state computations were triggered. The pfs from the sources in our experiments were such that, at each k, feasibility of the problem was ensured (see Lemma 1). Following [20], we also implemented an utility function that restricts calculations of the agent only to the road links that can be reached in T time steps (rather than through the whole state space/map). With this feature, in our experiments the algorithm took on average approximately half a second to output a behavior, less than the typical time taken to drive through a road link.[1] Finally, the pfs of the sources were obtained via built-in routing functions in SUMO and we added noise to the routes so that Assumption 2 would be fulfilled (for each road link, $\mathcal{S}(\pi)$ is the set of outgoing links). See our github for the details.

C. Experimental Results

Simulation results: first, we benchmarked the performance obtained by Algorithm 1 against these from the algorithm in [3], [4], termed as crowdsourcing algorithm in what follows. To this aim, we considered Scenario 1 and performed two sets of 10 simulations. In the first set of experiments, Algorithm 1 was used to determine the behavior of cars on campus (note that Assumption 1 is fulfilled). In the second set of simulations, the cars instead used the crodwourcing algorithm. Across the simulations, we recorded the number of cars that the algorithms were able to park. The results are illustrated in Figure 3 (top panel). The figure shows that the crowdsourcing algorithm was not able to park all the cars within the simulation. This was instead achieved by Algorithm 1, which outperformed the algorithm from [3], [4]. To further quantify the performance, we also computed the average time spent by a car looking for a parking space after it enters the simulation (ATTP: average time-to-parking). Across the simulations, the ATTP for the algorithm in [3] was of 224.74 ± 19.67, while for Algorithm 1 it was of 151.32 ± 30.59 (first quantities are means, second quantities are standard deviations). That is, Algorithm 1 yielded an average improvement of 32.7% in the ATTP. Then, we simulated the three cases of Scenario 2 to verify that Algorithm 1 can effectively regulate access through specific links. The constraints for the three cases of Scenario 2 were given as an input to the algorithm and in Figure 3 (bottom panel) the optimal solution $\pi^*(x_k \mid x_{k-1} = l_r)$ is shown. The figure shows that the optimal solution indeed fulfills the constraints (the road link l_r is in Figure 1).

HiL results: we deployed Algorithm 1 on a real vehicle using the HiL platform and validated its effectiveness in Scenario 2 (C): the target behavior of the agent would make the real car reach the *Terminal* parking but this route is forbidden. What we observed in the experiment was that, once the car entered in the campus, this was re-routed towards the *Biblioteca* parking. The re-routing was an effect of Algorithm 1 computing the rightmost pf in Figure 3 (bottom panel). A video of the HiL experiment is available on our github. The video shows that the algorithm is suitable for real car operation: it would run smoothly during the drive, providing feedback to the driver on the vehicle. Figure 4 shows the car's route recorded during the experiment.

VI. CONCLUSIONS

We considered the problem of designing agents able to compute optimal decisions by re-using data from multiple sources to solve tasks involving: (i) tracking a desired behavior while minimizing an agent-specific cost; (ii) satisfying certain safety constraints. After formulating the control problem, we showed that this is convex under a mild condition and computed the optimal solution. We turned the results in an algorithm and used it to design an intelligent parking system. We evaluated the algorithm via in-silico and in-vivo experiments with real vehicles/drivers. All experiments confirmed the effectiveness of the algorithm and its suitability for in-car operation. Besides considering the use of other divergences in the cost and deploying our results in more complex urban scenarios that would use data from pedestrians and sensors on-board vehicles, our future research will involve devising mechanisms for the composition of policies for the tasks with actuation constraints in [22].

REFERENCES

[1] B. M. Lake *et al.*, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.
[2] E. Crisostomi *et al.*, *Analytics for the sharing economy: Mathematics, Engineering and Business perspectives*. Springer, 2020.
[3] G. Russo, "On the crowdsourcing of behaviors for autonomous agents," *IEEE Cont. Sys. Lett.*, vol. 5, pp. 1321–1326, 2020.
[4] É. Garrabé and G. Russo, "On the design of autonomous agents from multiple data sources," *IEEE Cont. Sys. Lett.*, vol. 6, pp. 698–703, 2021.
[5] L. Li, C. De Persis, P. Tesi, and N. Monshizadeh, "Data-based transfer stabilization in linear systems," 2022. [Online]. Available: https://arxiv.org/abs/2211.05536
[6] J. Coulson, J. Lygeros, and F. Dörfler, "Data-enabled predictive control: In the shallows of the DeePC," in *European Control Conference*, 2019, pp. 307–312.
[7] H. J. van Waarde, J. Eising, H. L. Trentelman, and M. K. Camlibel, "Data informativity: a new perspective on data-driven analysis and control," *IEEE Trans. Automatic Control*, vol. 65, pp. 4753–4768, 2020.
[8] C. De Persis and P. Tesi, "Formulas for data-driven control: Stabilization, optimality, and robustness," *IEEE Trans. Automatic Control*, vol. 65, pp. 909–924, 2020.
[9] F. Celi and F. Pasqualetti, "Data-driven meets geometric control: Zero dynamics, subspace stabilization, and malicious attacks," *IEEE Cont. Sys. Lett.*, vol. 6, pp. 2569–2574, 2022.
[10] U. Rosolia and F. Borrelli, "Learning model predictive control for iterative tasks. A data-driven control framework," *IEEE Trans. Automatic Control*, vol. 63, pp. 1883–1896, 2018.

[1]this duration was measured between the moment where the GUI shows the car merging on a new link and the moment where new directions are displayed. The simulation was run on a modern laptop.

[11] K. P. Wabersich and M. N. Zeilinger, "Bayesian model predictive control: Efficient model exploration and regret bounds using posterior sampling," ser. Proc. of ML Research, vol. 120, 2020, pp. 455–464.

[12] J. Berberich, J. Kohler, M. A. Muller, and F. Allgower, "Data-driven model predictive control with stability and robustness guarantees," *IEEE Trans. Aut. Contr.*, vol. 66, pp. 1702–1707, 2021.

[13] G. Baggio, V. Katewa, and F. Pasqualetti, "Data-driven minimum-energy controls for linear systems," *IEEE Cont. Sys. Lett.*, vol. 3, pp. 589–594, 2019.

[14] Émiland Garrabé and G. Russo, "Probabilistic design of optimal sequential decision-making algorithms in learning and control," *Annual Rev, in Control*, vol. 54, pp. 81–102, 2022.

[15] N. Cammardella, A. Busic, Y. Ji, and S. P. Meyn, "Kullback-Leibler-Quadratic optimal control of flexible power demand," in *IEEE 58th Conference on Decision and Control*, 2019, pp. 4195–4201.

[16] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games.* Cambridge University Press, 2006.

[17] M. Cutler, T. J. Walsh, and J. P. How, "Real-world reinforcement learning via multifidelity simulators," *IEEE Trans. on Robotics*, vol. 31, pp. 655–671, 2015.

[18] V. B. Mountcastle, "The columnar organization of the neocortex." *Brain*, vol. 120, pp. 701–722, Apr 1997.

[19] W. Griggs *et al.*, *A Vehicle-in-the-Loop Emulation Platform for Demonstrating Intelligent Transportation Systems.* Cham: Springer International Publishing, 2019, pp. 133–154.

[20] É. Garrabé and G. Russo, "CRAWLING: a Crowdsourcing Algorithm on Wheels for Smart Parking," 2022, preprint submitted to Scientific Reports. [Online]. Available: https://arxiv.org/abs/2212.02467

[21] Pablo Alvarez Lopez and others, "Microscopic traffic simulation using SUMO," in *21st IEEE International Conference on Intelligent Transportation Systems*, 2018, pp. 2575–2582.

[22] D. Gagliardi and G. Russo, "On a probabilistic approach to synthesize control policies from example datasets," *Automatica*, vol. 137, p. 110121, 2022.
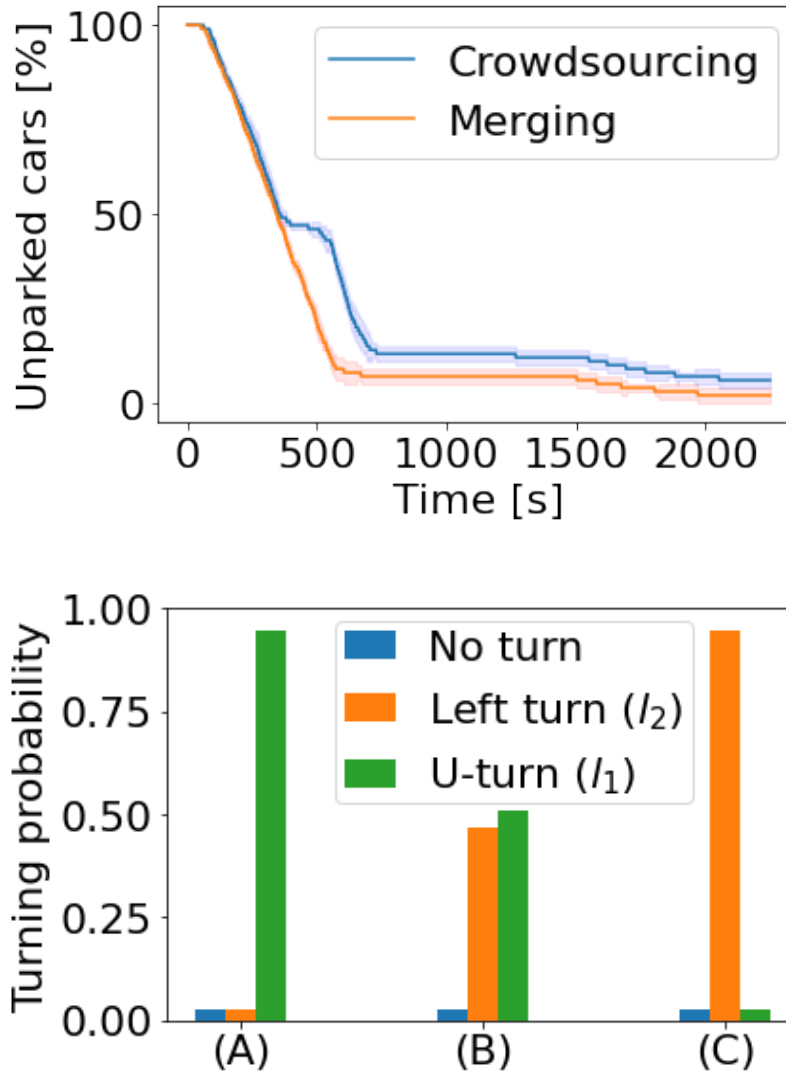
Fig. 3: Top: unparked cars over time for crowdsourcing and Algorithm 1. Solid lines are means across the simulations, shaded areas are confidence intervals (one standard deviation). Bottom: $\pi^*(x_k \mid x_{k-1} = l_r)$ for the three cases of Scenario 2. The pfs satisfy the constraints. Link definitions in Figure 1.

We report here an investigation of how the time taken by Algorithm 1 changes as a function of the number of sources, S, and time horizon T. This time is a crucial aspect to investigate whether the approach we propose would scale to more complex urban scenarios than the one presented in Section V, which would e.g., include more parking locations (note that these are seen as links by Algorithm 1) and more complex road networks together with more sources that the agent could use to determine its optimal behavior. The time Algorithm 1 takes to output a behavior depends on the number of available sources, the time horizon T and the number of links accessible to the car within the time horizon. We analyze the computation time w.r.t. the

To investigate Algorithm 1's computation time, we considered the same implementation as in Section V-B and ran the algorithm by varying the receding horizon window between 0 and 5 time steps, and the number of sources available to the agent between 1 and 6. For this, additional sources were taken from [20], where more sources were used to implement the algorithm from [4]. For each pair of these parameters, we measured the time taken by Algorithm 1 to output a behavior, by running the algorithm over each link in the network on a standard computer and taking the average of such times. This gives a fair estimate of the average runtime, as the amount of states considered depends on the density of the connections in the neighborhood of each link.

The results of this numerical investigation are shown in Figure 5. The figure shows that the highest computation time is about one second, which appears to be suitable for our reference application.

Fig. 4: Route of the real vehicle. The continuous line shows the GPS position during the HiL experiment (map from OpenStreetMaps).
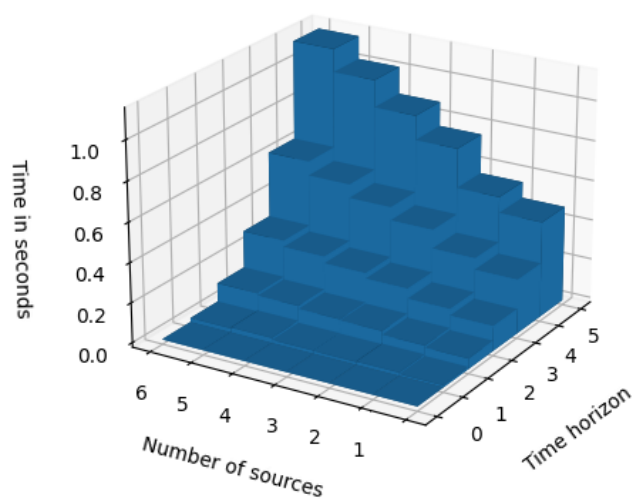
Fig. 5: Computation time as a function of time horizon and number of sources