This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2020.3018029, IEEE Transactions on Automatic Control

Synthesis of Provably Correct Autonomy Protocols for Shared Control

Murat Cubuktepe, Nils Jansen, Mohammed Alshiekh, Ufuk Topcu

***Abstract*—We synthesize shared control protocols subject to probabilistic temporal logic specifications. Specifically, we develop a framework in which a human and an autonomy protocol can issue commands to carry out a certain task. We blend these commands into a joint input to a robot. We model the interaction between the human and the robot as a Markov decision process representing the shared control scenario. Using inverse reinforcement learning, we obtain an abstraction of the human's behavior. We use randomized strategies to account for randomness in human's decisions, caused by factors such as the complexity of the task specifications or imperfect interfaces. We design the autonomy protocol to ensure that the resulting robot behavior satisfies given safety and performance specifications in probabilistic temporal logic. Additionally, the resulting strategies generate behavior as similar to the behavior induced by the human's commands as possible. We solve the underlying problem efficiently using quasiconvex programming. Case studies involving autonomous wheelchair navigation and unmanned aerial vehicle mission planning showcase the applicability of our approach.**

I. Introduction

In shared control, a robot executes a task to accomplish the goals of a human operator while adhering to additional safety and performance requirements. Applications of such human-robot interaction include remotely operated semi-autonomous wheelchairs [13], robotic teleoperation [17], and human-in-the-loop unmanned aerial vehicle mission planning [9]. A human operator issues a command through an input interface, which maps the command directly to an action for the robot. The problem is that a sequence of such actions may fail to accomplish the task at hand, due to limitations of the interface or failure of the human operator in comprehending the complexity of the problem. Therefore, a so-called *autonomy protocol* provides assistance for the human in order to complete the task according to the given requirements.

At the heart of the shared control problem is the design of an autonomy protocol. In the literature, there are two main directions, based on either *switching* the control authority between human and autonomy protocol [26], or on *blending* their commands towards joined inputs for the robot [7], [16].

One approach to switching the authority first determines the desired goal of the human operator with high confidence and then assists towards exactly this goal [8], [19]. In [12], switching the control authority between the human and autonomy protocol ensures the satisfaction of specifications that are formally expressed in temporal logic. In general, switching of authority may cause a decrease in human's satisfaction, who usually prefers to retain as much control as possible [18].

Blending incorporates providing an alternative command in addition to the command of the human operator.B Both commands are then blended to form a joined input for the robot to introduce a more flexible trade-off between the human's control authority and the level of autonomous assistance. A *blending function* determines the emphasis that is put on the autonomy protocol in the blending, that is, regulating the amount of assistance provided to the human [6], [7], [21]. Switching of authority can be seen as a special case of blending, as the blending function may assign full control to the autonomy protocol or the human. However, none of the existing blending approaches provide *formal correctness* guarantees that go beyond statistical confidence bounds. Correctness here refers to ensuring safety and optimizing performance according to the given requirements. Our goal is to design an autonomy protocol that admits formal correctness while rendering the robot behavior as close to the human's commands as possible, which is shown to enhance the human experience.

A human may be uncertain about which command to issue in order to accomplish a task. Moreover, a typical interface used to parse human's commands, such as a brain-computer interface, is inherently imperfect. To capture such uncertainties and imperfections in the human's decisions, we introduce *randomness* to the commands issued by humans. It may not be possible to blend two different deterministic commands. If the human's command is "up" and the autonomy protocol's command is "right", we cannot blend these two commands to obtain another deterministic command. By introducing randomness to the commands of the human and the autonomy protocol, we ensure that the blending is always well-defined.

Take as an example of a scenario involving a semi-autonomous wheelchair [13] whose navigation has to account for a randomly moving autonomous vacuum cleaner, see Fig. 1. The wheelchair needs to navigate to the exit of a room, and the vacuum cleaner moves according to a probabilistic transition function. The task of the wheelchair is to reach the exit gate while not crashing into the vacuum cleaner. The human may not fully perceive the motion of the vacuum cleaner. Note that the human's commands, depicted with the solid green line in Fig I, may cause the wheelchair to crash into the vacuum cleaner. The autonomy protocol provides another set of commands, which is indicated by the solid red line in Fig I, to carry out the task safely without crashing. However, the autonomy protocol's commands deviate highly from the commands of the human. The two sets of commands are then blended into a new set of commands, depicted using the dashed red line in Fig I. The blended commands perform the task safely while generating

M. Cubuktepe and U. Topcu are with the Department of Aerospace Engineering and Engineering Mechanics, University of Texas at Austin, 201 E 24th St, Austin, TX 78712, USA. Nils Jansen is with the Department of Software Science, Radboud University Nijmegen, Comeniuslaan 4, 6525 HP Nijmegen, the Netherlands. Mohammed Alshiekh was with the Institute for Computational Engineering and Sciences, University of Texas at Austin, 201 E 24th St, Austin, TX 78712, USA. email:({mcubuktepe,malshiekh,utopcu}@utexas.edu, n.jansen@science.ru.nl). This work was partially supported by grants NSF CNS-1836900, and NSF 1652113.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2020.3018029, IEEE Transactions on Automatic Control

behavior as similar to the behavior induced by the human's commands as possible.

We model the behavior of the robot as a Markov decision process (MDP) [25], which captures the robot's actions inside a potentially stochastic environment. Problem formulations with MDPs typically focus on maximizing an expected reward (or minimizing the expected cost). However, such formulations may not be sufficient to ensure safety or performance guarantees in a task that includes a human operator. Recently, it was shown that a reward structure is not sufficient to capture temporal logic constraints in general [15]. We design the autonomy protocol such that the resulting robot behavior satisfies probabilistic temporal logic specifications. Such verification problems have been extensively studied for MDPs [2], and mature tools exist for efficient verification [20], [5].

In what follows, we call a formal interpretation of a sequence of the human's commands the *human strategy*, and the sequence of commands issued by the autonomy protocol the *autonomy strategy*. In [16], we formulated the problem of designing the autonomy protocol as a *nonlinear programming problem*. However, solving nonlinear programs is generally intractable [3]. Therefore, we proposed a greedy algorithm that iteratively *repairs* the human strategy such that the specifications are satisfied without guaranteeing an optimal solution, based on [24]. Here, we propose an alternative approach for the blending of the two strategies. We follow the approach of repairing the strategy of the human to compute an autonomy protocol. We ensure that the resulting robot behavior induced by the repaired strategy deviates minimally from the human strategy, and satisfies safety and performance properties given in temporal logic specifications. We formally define the problem as a *quasiconvex optimization problem*, which can be solved efficiently by checking the feasibility of a number of convex optimization problems [4].

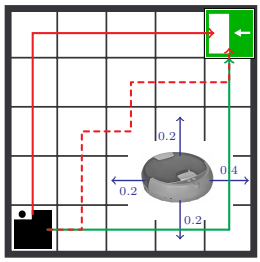


Fig. 1. A wheelchair in a shared control setting with the human's perspective.

The question remains on how to obtain the human strategy in the first place. It may be unrealistic that a human can provide the strategy for an MDP that models a realistic scenario. To this end, we create a virtual simulation environment that captures the behavior of the MDP. We ask humans to participate in two case studies to collect data about typical human behavior. We use *inverse reinforcement learning* to get a formal interpretation as a strategy based on human's inputs [1], [28]. We model a typical shared control scenario based on an autonomous wheelchair navigation [13] in our first case study. In our second case study, we consider an unmanned aerial vehicle mission planning scenario, where the human operator is to patrol certain regions while keeping away from enemy aerial vehicles.

In summary, the main contribution of this paper is to efficiently synthesize an autonomy protocol such that the resulting blended or repaired strategy meets all given specifications while only minimally deviating from the human strategy. We present a new technique based on quasiconvex programming, which can be solved efficiently using convex optimization [4].

Organization. We introduce all formal foundations that we need in Section II. We provide an overview of the shared control problem in Section III. We present the *shared control synthesis problem* and provide a solution based on convex optimization in Section IV. We indicate the applicability and scalability of our approach on experiments in Section V and draw a conclusion and critique of our approach in Section VI.

II. Preliminaries

In this section, we introduce the required formal models and specifications that we use to synthesize the autonomy protocol, and we give a short example illustrating the main concepts.

A. Markov Decision Processes

A *probability distribution* over a finite set X is a function $\mu\colon X \to [0,1] \subseteq \mathbb{R}$ with $\sum_{x\in X} \mu(x) = \mu(X) = 1$. The set $Distr(X)$ denotes all probability distributions over X.

Definition 1 (MDP)**.** *A* Markov decision process (MDP) $\mathcal{M} = (S, s_I, Act, \mathcal{P})$ *has a finite set* S *of states, an initial state* $s_I \in S$*, a finite set* Act *of actions, a transition probability function* $\mathcal{P}\colon S \times Act \to Distr(S)$.

MDPs have *non-deterministic choices* of actions at states; the successors are determined *probabilistically* via associated probability distributions. A *cost function* $C\colon S \times Act \to \mathbb{R}_{\geq 0}$ associates cost to state-action pairs. If there is only a single action at each state, the MDP reduces to a *Markov chain (MC)*. We use *strategies* to resolve the choices of actions in order to define a probability and expected cost measure for MDPs.

Definition 2 (Strategy)**.** *A* memoryless and randomized strategy *for an MDP* $\mathcal{M}$ *is a function* $\sigma\colon S \to Distr(Act)$*. The set of all strategies over* $\mathcal{M}$ *is* $Str^{\mathcal{M}}$.

Resolving all the nondeterminism for an MDP $\mathcal{M}$ with a strategy $\sigma \in Str^{\mathcal{M}}$ yields an *induced Markov chain* $\mathcal{M}^\sigma$.

Definition 3 (Induced MC)**.** *For an MDP* $\mathcal{M} = (S, s_I, Act, \mathcal{P})$ *and strategy* $\sigma \in Str^{\mathcal{M}}$*, the* MC induced by $\mathcal{M}$ and σ *is* $\mathcal{M}^\sigma = (S, s_I, Act, \mathcal{P}^\sigma)$*, where*

$$\mathcal{P}^\sigma(s,s') = \sum\nolimits_{\alpha\in Act(s)} \sigma(s,\alpha)\cdot\mathcal{P}(s,\alpha,s') \text{ for all } s,s' \in S.$$

In our solution, we use the occupancy measure of a strategy to compute an autonomy protocol, which is introduced below.

Definition 4 (Occupancy Measure)**.** *[25] The occupancy measure* x_σ *of a strategy* σ *for an MDP* $\mathcal{M}$ *is defined as*

$$x_\sigma(s,\alpha) = \mathbb{E}\left[\sum\nolimits_{t=0}^{\infty} \mathcal{P}^\sigma(s_t = s, \alpha_t = \alpha | s_0 = s_I)\right], \quad (1)$$

where s_t *and* α_t *denote the state and action in* $\mathcal{M}$ *at time* t.

The occupancy measure $x_\sigma(s,\alpha)$ is the expected number of times to take action α at state s under the strategy σ, and the strategy gives the probability of taking action α at state s.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2020.3018029, IEEE Transactions on Automatic Control

B. Specifications

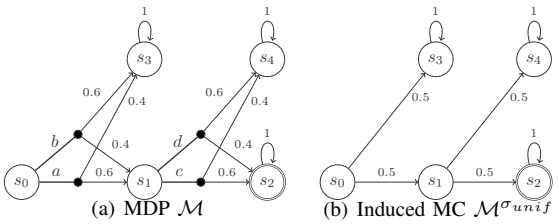
For an MDP $\mathcal{M}$, the *reachability specification* $\varphi_r = \mathbb{P}_{\geq\beta}(\Diamond T)$ states that a set $T \subseteq S$ of *target states* is reached with probability at least $\beta \in [0,1]$. The *synthesis problem* is to find one particular strategy σ for an MDP $\mathcal{M}$ such that given a reachability specification φ_r and a threshold $\beta \in [0,1]$, the induced MC $\mathcal{M}^\sigma$ satisfies $\mathbb{P}_{\mathcal{M}^\sigma}(s_I \models \varphi_r) \geq \beta$, which implies that the strategy σ satisfies the specification φ_r. We note that linear temporal logic specifications can be reduced to reachability specifications [2], therefore we omit a detailed introduction. We also consider expected cost properties $\varphi_c = \mathbb{E}_{\leq\kappa}(\Diamond G)$, that restricts the expected cost to reach the set $G \subseteq S$ of goal states by an upper bound κ. The expected cost of a strategy σ is given by

$$\sum\nolimits_{s\in S}\sum\nolimits_{\alpha\in Act(s)} x_\sigma(s,\alpha)C(s,\alpha). \tag{2}$$

Intuitively, for strategy σ the cost and the expected number of taking action α at state s are multiplied. This multiplication is summed up for all states and actions.

Example 1. *Fig. 2(a) depicts an MDP $\mathcal{M}$ with initial state s_0. In state s_0, the available actions are a and b. Similarly for state s_1, the two available actions are c and d. If action a is selected in state s_0, the agent transitions to s_1 and s_3 with probabilities 0.6 and 0.4.*

For a reachability specification $\varphi_r = \mathbb{P}_{\geq 0.30}(\Diamond s_2)$, the deterministic strategy $\sigma_1 \in Str^{\mathcal{M}}$ with $\sigma_1(s_0,b) = 1$ and $\sigma_1(s_1,s) = 1$ induces a probability of 0.16 to reach s_2. Therefore, the specification is not satisfied, see the induced MC in Fig. 2(b). Likewise, the randomized strategy $\sigma_{unif} \in Str^{\mathcal{M}}$ with $\sigma_{unif}(s_0,a) = \sigma_{unif}(s_0,b) = 0.5$ and $\sigma_{unif}(s_1,c) = \sigma_{unif}(s_1,d) = 0.5$ violates the specification, as the probability of reaching s_2 is 0.25. However, the deterministic strategy $\sigma_{safe} \in Str^{\mathcal{M}}$ with $\sigma_{safe}(s_0,a) = 1$ and $\sigma_{safe}(s_1,c) = 1$ induces a probability of 0.36, thus $\sigma_{safe} \models \varphi_r$.



(a) MDP $\mathcal{M}$ (b) Induced MC $\mathcal{M}^{\sigma_{unif}}$

Fig. 2. MDP $\mathcal{M}$ with target state s_2 and induced MC for strategy σ_{unif}

III. Conceptual Description of Shared Control

We now detail the general shared control concept adopted in this paper and state the formal problem. As inputs, we have a set of task specifications, a model $\mathcal{M}$ for the robot behavior, and a blending function b. The given robot task is described by certain reachability and expected cost specifications $\varphi = \varphi_1 \wedge \varphi_2 \ldots \wedge \varphi_n$. For example, it may not be safe to take the shortest route because there may be too many obstacles in that route. In order to satisfy performance considerations, the robot should prefer to take the shortest route possible while not violating the safety specifications. We model the behavior of the robot inside a stochastic environment as an MDP $\mathcal{M}$.

It may be unrealistic that a human grasps an MDP that models a realistic shared control scenario. Indeed, a human will likely have difficulties interpreting a large number of possibilities and the associated probability of paths and payoffs [11]. We obtain a human strategy as an abstraction of a sequence of human's commands using inverse reinforcement learning [1], [28]. Specifically, we compute a formal human strategy σ_h based on specific inputs of a human.

The *shared control synthesis problem* is then computing a repaired strategy σ_{ha} such that it holds $\sigma_{ha} \models \varphi$ while *deviating minimally* from σ_h. The deviation between the human strategy σ_h and the repaired strategy σ_{ha} is measured by the maximal difference between the two strategies in each state of the MDP. We state the problem that we study as follows.

Problem 1. *Let $\mathcal{M}$ be an MDP, φ be an LTL specification, σ_h be a human strategy, and β be a constant. Synthesize a repaired strategy $\sigma_{ha} \in Str^{\mathcal{M}}$ that solves the following problem.*

$$\underset{\sigma_{ha}\in Str^{\mathcal{M}}}{\text{minimize}} \quad \max_{s\in S,\alpha\in Act} |\sigma_h(s,\alpha) - \sigma_{ha}(s,\alpha)| \tag{3}$$

$$\text{subject to} \quad \mathbb{P}_{\mathcal{M}^{\sigma_{ha}}}(s_I \models \varphi) \geq \beta. \tag{4}$$

Using the repaired strategy, and the blending function b, we compute the automomy strategy σ_a. The blending function reflects preference over the human strategy or the autonomy strategy in all states of the MDP. At runtime, we can then blend commands of the human and the autonomy strategy. The resulting "blended" commands will induce the same behavior as the blended strategy σ_{ha}, and the specifications are satisfied. Note that blending commands at runtime according to predefined blending function and autonomy protocol requires a linear combination of real values and is thus very efficient.

IV. Synthesis of the Autonomy Protocol

In this section, we describe our approach to synthesize an autonomy protocol for the shared control synthesis problem. We start with the concepts of strategy blending and strategy repair. We then show how to synthesize a repaired strategy that deviates minimally from the human strategy based on quasiconvex programming. Finally, we discuss how to include additional specifications and discuss other measures for the human and the repaired strategy that induce a similar behavior.

A. Strategy blending

Given the human strategy σ_h and the autonomy strategy σ_a, a blending function computes a weighted composition of the two strategies by favoring one or the other strategy in each state of the MDP at runtime [17], [6], [7]. As a design choice, the blending function weighs the confidence in the human strategy and autonomy strategy at each state of the MDP.

Definition 5 (Linear blending)**.** *Given the MDP $\mathcal{M} = (S, s_I, Act, \mathcal{P})$, two memoryless strategies σ_h, σ_a, and a* blending function $b\colon S \to [0,1]$, *the blended strategy $\sigma_{ha} \in Str^{\mathcal{M}}$ for all states $s \in S$, and actions $\alpha \in Act$ is*

$$\sigma_{ha}(s,\alpha) = b(s)\cdot\sigma_h(s,\alpha) + (1-b(s))\cdot\sigma_a(s,\alpha). \tag{5}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2020.3018029, IEEE Transactions on Automatic Control

For each $s \in S$, the value of $b(s)$ represents the "weight" of σ_h at s, meaning how much emphasis $b(s)$ puts on the human strategy at state s. For instance, if $b(s) = 0.1$, we infer that the blended strategy in a state s puts more emphasis on σ_a. Note that in our experiments, we design the blending function by making use of the inherent verification result. For each state s, we will compute the probability of satisfying the objective under the human strategy σ_h. If that probability is too low, we will put higher emphasis on the autonomy.

B. Solution to the shared control synthesis problem

We propose an algorithm to solve the shared control synthesis problem. Our solution is based on quasiconvex programming, which can be solved by checking the feasibility of a number of convex optimization problems. We show that the result is the repaired strategy as in Problem 1. The strategy satisfies the specifications and deviates minimally from the human strategy. We use that result to compute the autonomy strategy σ_a.

1) Dual linear programming formulation for MDPs: In this section, we recall the dual LP formulation to compute a strategy that maximizes the probability of satisfying a reachability specification φ_r in an MDP [25], [10].

The variables of the dual LP formulation are following:

- $x_{\sigma_{ha}}(s, \alpha) \in [0, \infty)$ for each state $s \in S \setminus T$ and action $\alpha \in Act$ defines the occupancy measure of a state-action pair for the strategy σ_{ha}, i.e., the expected number of times of taking action α in state s.
- $x_{\sigma_{ha}}(s) \in [0, 1]$ for each state $s \in T$ defines the probability of reaching a state $s \in T$.

$$\text{maximize} \quad \sum\nolimits_{s \in T} x_{\sigma_{ha}}(s) \tag{6}$$

$$\text{subject to} \quad \forall s \in S \setminus T. \tag{7}$$

$$\sum_{\alpha \in Act} x_{\sigma_{ha}}(s, \alpha) = \sum_{s' \in S \setminus T} \sum_{\alpha \in Act} \mathcal{P}(s', \alpha, s) x_{\sigma_{ha}}(s', \alpha) + \alpha_s$$

$$\forall s \in T.$$

$$x_{\sigma_{ha}}(s) = \sum_{s' \in S \setminus T} \sum_{\alpha \in Act} \mathcal{P}(s', \alpha, s) x_{\sigma_{ha}}(s', \alpha) + \alpha_s \tag{8}$$

$$\sum\nolimits_{s \in T} x_{\sigma_{ha}}(s) \geq \beta \tag{9}$$

where $\alpha_s = 1$ if $s = s_I$ and $\alpha_s = 0$ if $s \neq s_I$. The constraints in (7) and (8) ensure that the expected number of times transitioning to a state $s \in S$ is equal to the expected number of times to take action α that transitions to a different state $s' \in S$. The constraint in (9) ensures that the specification φ_r is satisfied with a probability of at least β. We determine the states with probability 0 to reach T by a preprocessing step on the underlying graph of the MDP. To ensure that the variables $x_{\sigma_{ha}}(s)$ encode the actual probability of reaching a state $s \in T$, we then set the variables of the states with probability 0 to reach T to zero.

For any optimal solution $x_{\sigma_{ha}}$ to the LP in (6)–(9),

$$\sigma_{ha}(s, \alpha) = \frac{x_{\sigma_{ha}}(s, \alpha)}{\sum_{\alpha' \in Act} x_{\sigma_{ha}}(s, \alpha')} \tag{10}$$

is an optimal strategy, and $x_{\sigma_{ha}}$ is the occupancy measure of σ_{ha}, see [25] and [10] for details.

2) Strategy repair using quasiconvex programming: Given σ_h, the aim of the autonomy protocol is to compute the blended strategy, or the repaired strategy σ_{ha} that induces a similar behavior to the human strategy and yet satsifies the specifications. We compute the repaired strategy by *repairing* the human strategy resulting in the following formulation:

Lemma 1. *The shared control synthesis problem can be formulated as the following nonlinear programming program with following variables:*

- $x_{\sigma_{ha}}$ as defined for the optimization problem in (6)–(9).
- $\hat{\delta} \in [0, 1]$ gives the maximal deviation between the human strategy σ_h and the repaired strategy σ_{ha}.

$$\text{minimize} \quad \hat{\delta} \tag{11}$$

$$\text{subject to} \ (7), \ (8), (9), \ \text{and} \tag{12}$$

$$\forall s \in S \setminus T. \, \forall \alpha \in Act \tag{13}$$

$$|x_{\sigma_{ha}}(s, \alpha) - \sum_{\alpha' \in Act} x_{\sigma_{ha}}(s, \alpha') \sigma_h(s, \alpha')| \leq \hat{\delta} \sum_{\alpha'' \in Act} x_{\sigma_{ha}}(s, \alpha'').$$

Proof. For any solution to the optimization problem above, the constraints in (12) ensure that the strategy computed by (10) satisfies the specification. We now show that by minimizing $\hat{\delta}$, we minimize the maximal deviation between the human strategy and the repaired strategy. We perturb σ_h to σ_{ha} by

$$\forall s \in S \setminus T. \alpha \in Act. \quad \sigma_{ha}(s, \alpha) = \sigma_h(s, \alpha) + \delta(s, \alpha).$$

where $\delta(s, \alpha)$ for $s \in S, \alpha \in Act$ is an perturbation function subject to σ_{ha} being a well-defined strategy. Using (10), we reformulate the above constraint into

$$\forall s \in S \setminus T. \alpha \in Act. \tag{14}$$

$$x_{\sigma_{ha}}(s, \alpha) = \sum\nolimits_{\alpha' \in Act} \left(x_{\sigma_{ha}}(s, \alpha') \left(\sigma_h(s, \alpha') + \delta(s, \alpha') \right) \right).$$

Since we are interested in minimizing the maximal deviation, we assign a common variable $\hat{\delta} \in [0, 1]$ for all state-action pairs in $\mathcal{M}$ to put an upper bound on the deviation by

$$\forall s \in S \setminus T. \alpha \in Act. \tag{15}$$

$$|x_{\sigma_{ha}}(s, \alpha) - \sum_{\alpha' \in Act} x_{\sigma_{ha}}(s, \alpha') \sigma_h(s, \alpha')| \leq \hat{\delta} \sum_{\alpha'' \in Act} x_{\sigma_{ha}}(s, \alpha'').$$

Therefore, by minimizing $\hat{\delta}$ subject to the constraints in (12)–(13) ensures that σ_{ha} deviates minimally from σ_h. □

The constraint in (15) is a quadratic and nonconvex constraint due to multiplication of $\hat{\delta}$ and x_{ha}. However, we show that the problem in (11)–(13) is a quasiconvex programming problem, which can be solved efficiently using bisection over $\hat{\delta}$ [4]. We give the formal definition of a quasiconvex function below.

Definition 6 (Quasiconvex functions [4])**.** *A function $f : \mathcal{D} \to \mathbb{R}$ is quasiconvex if its domain $\mathcal{D}$ and all its sublevel sets $\mathcal{D}_\mu := \{y \in \mathcal{D} \colon f(x) \leq \mu\}$ for $\mu \in \mathbb{R}$ are convex sets.*

Lemma 2. *The nonlinear programming problem in* (11)–(13) *is a quasiconvex programming problem.*

Proof. For a fixed $\hat{\delta}$, the set described by the inequality in (15) is convex, that is, the *sublevel sets* of the function are convex [4, Section 3.4]. Therefore, the constraint in (15) is quasiconvex

and the nonlinear programming problem in (11)–(13) is a quasiconvex programming problem (QCP). □

We solve the QCP in (11)–(13) by employing bisection over the variable $\hat{\delta}$. We initialize a lower and upper bound of the maximal deviation between the human strategy and the repaired strategy to 0 and 1 respectively. Then, we iteratively refine the bounds by solving a number of convex feasibility problems. A method to solve quasiconvex optimization problems is given in [4, Algorithm 4.1]. Our approach is given in Algorithm 1 based on Algorithm 4.1 in [4]. We now state the main result.

Algorithm 1: Bisection method to synthesize an optimal repaired strategy σ_{ha} for the shared control synthesis problem.

given $\mathcal{M} = (S, s_I, Act, \mathcal{P})$, σ_h, $l = 0, u = 1$, $\epsilon > 0$.
repeat
1. Set $\hat{\delta} = (l + u)/2$.
2. Solve the convex feasibility problem in (12)–(13).
3. **if** the problem in (12)–(13) is feasible, **then**

$$u := \hat{\delta},\ \sigma_{ha}(s,\alpha) = \frac{x_{\sigma_{ha}}(s,\alpha)}{\sum_{\alpha' \in Act} x_{\sigma_{ha}}(s,\alpha')}$$

else $l := \hat{\delta}$.
until $u - l \leq \epsilon$.

Theorem 1. *The repaired strategy σ_{ha} obtained from Algorithm 1 satisfies the task specifications and it deviates minimally from the human strategy σ_h, and is an optimal solution to the shared control synthesis problem.*

Proof. From a satisfying assignment to the constraints in (11)–(13), we compute a strategy that satisfies the specification using (10). Using Algorithm 1, we compute the repaired strategy σ_{ha} that deviates minimally from the human strategy σ_h up to ϵ accuracy in $\left\lceil \log_2(\frac{1}{\epsilon}) \right\rceil$ iterations. Therefore, Algorithm 1 computes an optimal strategy. □

Algorithm 1 computes the minimally deviating repaired strategy σ_{ha} that satisfies the LTL specification. In [16], we considered computing an autonomy protocol with a greedy approach, which requires solving possibly a large number of LPs to compute a feasible strategy that is not necessarily optimal. On the other hand, using Algorithm 1, we only need to check feasibility of a number of LPs that can be determined to compute an optimal strategy. After computing σ_{ha}, we compute the autonomy strategy σ_a according to the Definition 5.

Computationally, the most expensive step of the Algorithm 1 is checking the feasibility of the optimization problem in (12)–(13). The number of variables and constraints in the optimization problem are linear in the number of states and actions in $\mathcal{M}$, therefore, checking feasibility of the optimization problem can be done in time polynomial in the size of $\mathcal{M}$ with interior point methods [22]. Algorithm 1 terminates after $\left\lceil \log_2(\frac{1}{\epsilon}) \right\rceil$ iterations, therefore we can compute an optimal strategy up to ϵ accuracy in time polynomial in the size of $\mathcal{M}$.

3) Additional specifications: The QCP in (11)–(13) computes an optimal strategy for a single LTL specification φ. Suppose that we are given another reachability specification $\varphi_r = \mathbb{P}_{\geq\lambda}(\lozenge B)$ with $B \in S$. We can handle this specification by appending the constraint

$$\sum\nolimits_{s \in B} x_{\sigma_{ha}}(s) \geq \lambda \tag{16}$$

to the QCP in (11)–(13). The constraint in (16) ensures that the probability of reaching T is greater than λ.

We handle an *expected cost specification* $\mathbb{E}_{\leq\kappa}(\lozenge G)$ for $G \subseteq S$, by adding the constraint

$$\sum\nolimits_{s \in S \setminus (T \bigcup G)} \sum\nolimits_{\alpha \in Act} C(s,\alpha) x_{\sigma_{ha}}(s,\alpha) \leq \kappa \tag{17}$$

to the QCP in (11)–(13). The constraint in (17) ensures that the expected cost of reaching G is less than κ.

Algorithm 1 with addition of these constraints does run in time polynomial in the size of $\mathcal{M}$, provided that the number of additional objectives is in polynomial in the size of $\mathcal{M}$ as each objective adds another constraint to the QCP in (11)–(13).

4) Additional measures: We discuss additional measures that can be used to render the behavior between the human and the autonomy protocol similar based on the occupancy measure of a strategy. Instead of minimizing the maximal deviation between the human strategy and the repaired strategy, we can also minimize the maximal difference of occupancy measures of the strategies. We can minimize the maximal difference of occupancy measures by minimizing $||x_{\sigma_{ha}} - x_{\sigma_h}||_\infty$.

The occupancy measure of the human strategy can be computed by finding a feasible solution to the constraints in (12) for the induced MC $\mathcal{M}^{\sigma_{ha}}$. We can also minimize other convex norms of the occupancy measures of the human strategy and the repaired strategy, such as 1-norm or 2-norm.

V. Numerical Examples

We present two numerical examples that illustrate the efficacy of the proposed approach. In the first example, we consider a wheelchair scenario from Fig. 1. The goal in this scenario is to reach the target state while not crashing with the obstacle. In the second example, an unmanned aerial vehicle (UAV) mission is considered, where the objective is to survey certain regions while avoiding enemy agents.

We require an representation of the human's commands as a strategy to use our synthesis approach in a shared control scenario. We discuss how such strategies are obtained using inverse reinforcement learning and report on case study results.

A. Experimental setting

We give an overview of the workflow of the experiments in Fig. 3. In an simulation environment, we collect sample data from the human's commands. Based on these commands, we compute a human strategy σ_h using maximum-entropy inverse reinforcement learning (MEIRL) [28]. MEIRL maximizes the likelihood of demonstrated paths by learning a cost function and strategy directly from demonstrations. The key notion, intuitively, is that an agent acts to optimize an unknown cost function, which are assumed to be linear in the features, and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2020.3018029, IEEE Transactions on Automatic Control

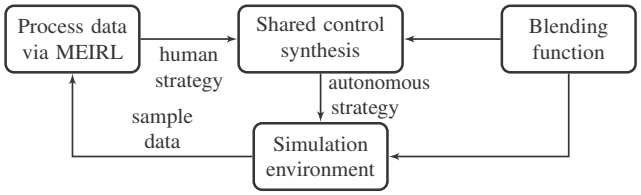


Fig. 3. The setting of the case study for the shared control simulation. We collect sample data from a simulation environment, and compute the human strategy using maximum-entropy inverse reinforcement learning (MEIRL).

the objective of MEIRL is to find cost weights that maximizes the likelihood of demonstrated paths.

After computing the human strategy, we synthesize the repaired strategy σ_{ha} using Algorithm 1. After synthesizing σ_{ha}, we compute the autonomous strategy σ_a using (5).

We model the wheelchair scenario inside an interactive `Python` environment. In the second scenario, we use the UAV simulation environment AMASE[1], developed at Air Force Research Laboratory. AMASE can be used to simulate multi-UAV missions. The graphical user interfaces of AMASE allow humans to send commands to one or multiple vehicles.

We use the model checker `PRISM` [20] to verify if the computed strategies satisfy the specification. We use the LP solver `Gurobi` [14] to check the feasibility of the LP problems that is given in Section IV. We also implemented the greedy approach for strategy repair in [16]. In this section, we refer to the procedure given by Algorithm 1 as QCP method, and the procedure from [16] as greedy method.

B. Data collection

We asked five participants to accomplish tasks in the wheelchair scenario. The goal is moving the wheelchair to a target cell in the gridworld while never occupying the same cell as the moving obstacle. Similarly, three participants performed the surveillance task in the AMASE environment.

From the data obtained from each participant, we compute an individual randomized human strategy σ_h via MEIRL. [17] uses inverse reinforcement learning to reason about the human's commands in a shared control scenario. However, they lack formal guarantees on the robot's execution.

In our setting, we denote each *sample* as one particular command of the participant, and we assume that the participant issues the command to satisfy the specification. Under this assumption, we can bound the probability of a possible deviation from the actual intent with respect to the number of samples using Hoeffding's inequality for the resulting strategy, see [27] for details. Using these bounds, we can determine the required number of commands to get an approximation of a typical human behavior. The probability of a possible deviation from the human behavior is given by $\mathcal{O}(\exp(-n\gamma^2))$, where n is the number of commands from the human and γ is the upper bound on the deviation between the probability of satisfying the specification with the true human strategy and the probability obtained by the strategy that is computed by inverse reinforcement learning. For example, to ensure an upper bound $\gamma = 0.05$ on the deviation of the probability of satisfying the specification with a probability of 0.99, we require 1060 demonstrations from the human.

[1]https://github.com/afrl-rq/OpenAMASE

TABLE I
SCALABILITY RESULTS FOR THE GRIDWORLD EXAMPLE.

States	Greedy time	δ_{G}	QCP time until δ_{G}	QCP time	δ_{QCP}
$2,304$	14.12	0.145	8.40	31.49	0.031
$14,400$	250.78	0.339	90.50	452.27	0.050
$40,000$	913.23	0.373	336.41	$1,682.05$	0.048

We design the blending function by assigning a lower weight to the human strategy at states where it yields a lower probability of reaching the target set. Using this function, we create the autonomy strategy σ_a and pass it (together with the blending function) back to the environment. Note that the repaired strategy σ_{ha} satisfies the specification, by Theorem 1. In runtime, we blend the human commands with the commands of the autonomy strategy σ_a.

C. Gridworld

The size of the gridworld in Fig. 1 is variable, and we generate a number of randomly moving (e.g., the vacuum cleaner) and stationary obstacles. An agent (e.g., the wheelchair) moves in the gridworld according to the commands from a human. For the gridworld scenario, we construct an MDP where the states represent the positions of the agent and the obstacles and the actions induce changes in the agent position.

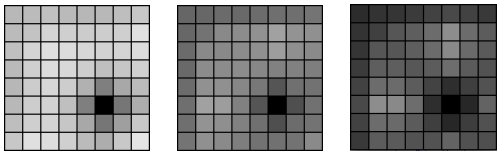
The safety specification states that the agent has to reach a target cell while not crashing into an obstacle with a certain probability $\beta \in [0, 1]$, formally $\mathbb{P}_{\geq\beta}(\neg\texttt{crash}\ \mathcal{U}\ \texttt{target})$.

First, we report results for one participant in a gridworld scenario with a 8×8 grid and one moving obstacle. The agent and the obstacle have four actions in all states, namely left, right, up and down. At each state, a transition to the chosen direction occurs with a probability of 0.7, and the agent transitions to each adjacent state in the chosen direction with a probability 0.15. If a transition to the wall occurs, the agent remains in the same state. The resulting MDP has 2304 states and 36864 transitions. We compute the human strategy using MEIRL where the features are the components of the cost function, for instance the distance to the obstacle and the goal.

We instantiate the safety specification with $\beta = 0.7$, which means the target should be reached with at least a probability of 0.7. The human strategy σ_h induces a probability of 0.546 to satisfy the specification. That is, it violates the specification.

We compute σ_{ha} using the greedy and the QCP approach, and both strategies satisfy the specification with a probability larger than β. On the one hand, the maximum deviation between σ_h and σ_{ha} is 0.15 with the greedy approach, which implies that the strategy of the human and the autonomy protocol deviates at most 15% for all states and actions. On the other hand, the maximum deviation between σ_h and σ_{ha} is 0.03 with the QCP approach. The results show that the QCP approach computes a repaired strategy that induces a more similar strategy to the human strategy compared to the LP approach.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TAC.2020.3018029, IEEE Transactions on Automatic Control



(a) Strategy σ_h (b) Strategy σ_{ah} (c) Strategy σ_a

Fig. 4. Graphical representation of the obtained human, blended, and autonomy strategy in the grid.

We give a graphical representation of the human strategy σ_h, repaired strategy σ_{ha}, and the autonomy strategy σ_a in Fig. 4. For each strategy, we indicate the average probability of safely reaching the target with the QCP approach. Note that the probability of reaching the target depends on the current position of the obstacle. Therefore, the probability for satisfying a specification could be higher or lower than shown in Fig. 4. The darkest cell in the each figure is the target state. In Fig. 4, the probability of reaching the target increases with a darker color, and black indicates a probability of 1 to reach the target. We observe that the human strategy induces a lower probability of reaching the target in most of the states compared to the repaired strategy. Note that the autonomy strategy induces a very high probability of reaching the target in each cell, but the autonomy strategy may not be similar to the human strategy.

To finally assess the scalability of our approach, consider Table I. We generated MDPs for different gridworlds with a different number of states and number of obstacles. We list the number of states in the MDP and the number of transitions. We report on the time that the synthesis process took with the greedy approach (labeled as "Greedy time") and QCP approach (labeled as "QCP time"), which includes the time of solving the LPs in the greedy method or QCPs measured in seconds. It also includes the model checking times using PRISM for the greedy approach. To represent the optimality of the synthesis, we list the maximal deviation between the repaired strategy and the human strategy for the greedy and QCP approach (labeled as "δ_{G}" and "δ_{QCP}"). We also include the time for the QCP approach (labeled as "QCP time until δ_{G}") when the computed maximal deviation of the QCP approach is less than δ_{G}. We observe that strategies obtained by the QCP approach yield autonomy strategies with less maximal deviation from the human strategy while having similar computation time as the greedy approach. Furthermore, the QCP approach is considerably faster than the greedy approach in all cases if we run the QCP approach until we reach an objective value that is similar to the greedy approach.

D. UAV mission planning

Similar to the gridworld scenario, we generate an MDP where states denote the position of the agent and the enemy agents in an AMASE scenario. Consider an example scenario in Fig. 5: The specification (or the mission) of the agent (blue UAV) is to keep surveilling the green regions (labeled as w_1, w_2, w_3) while avoiding restricted operating zones (labeled as "ROZ_1, ROZ_2") and enemy agents (purple and green UAVs). We asked the participants to visit the regions in a sequence, i.e., visiting the first region, then second, and then the third region. After visiting the third region, the task is to visit the first region. For example, if the last visited region is w_3, then the safety specification in this scenario is $\mathbb{P}_{\geq\beta}((\neg\texttt{crash} \wedge \neg\texttt{ROZ})\ \mathcal{U}\ \texttt{target})$, where ROZ is to visit the ROZ areas and target is visiting w_1.

TABLE II
RESULTS FOR DIFFERENT SPECIFICATION THRESHOLDS FOR THE PROBABILITY AND EXPECTED TIME IN THE AMASE EXAMPLE.

β	κ	Synthesis time	δ_{QCP}
0.7	20	827.37	0.380
0.7	40	749.14	0.126
0.7	80	722.81	0.054
0.9	20	888.29	0.598
0.9	40	795.98	0.163
0.9	80	732.41	0.100

We synthesize the autonomy protocol on the AMASE scenario with two enemy agents. The underlying MDP has 15625 states. The blending function and threshold $\beta = 0.7$ are same as in the grid-world example. The features to compute the human strategy are given by the distance to the closest ROZ, enemy agents, and the target region.

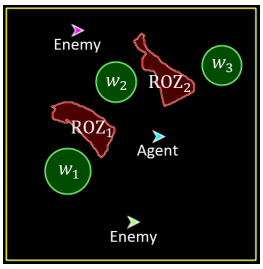


Fig. 5. Snapshot of a simulation using the AMASE simulator. The objective of the agent is to keep surveilling the green regions while avoiding enemy agents and restricted operating zones.

The human strategy σ_h violates the specification with a probability of 0.496. Again, we compute the repaired strategy σ_{ha} with the greedy and the QCP approach. Both strategies satisfy the specification. On the one hand, the maximum deviation between σ_h and σ_{ha} is 0.418 with the greedy approach, which means the strategies of the human and the autonomy protocol are significantly different in some states of the MDP. On the other hand, the QCP approach yields a repaired strategy σ_{ha} that is more similar to the human strategy σ_h with a maximum deviation of 0.038. The time of the synthesis procedure with the LP approach is 481.31 seconds and the computation time with the QCP approach is 749.18 seconds, showing the trade-offs between the greedy approach and the QCP approach. We see that the greedy approach can compute a feasible solution slightly faster, however the resulting blended strategy may be less similar to the human strategy compared to the QCP approach.

To assess the effect of changing specifications, we use a different threshold $\beta = 0.9$. The greedy approach did not terminate within one hour, and could not find a repaired strategy that satisfies the specification after 45 iterations. We compute a repaired strategy σ_{ha} using the QCP approach with a maximum deviation of 0.093. The computation time with the QCP approach is 779.81 seconds, showing that the QCP approach does not take significantly more time even with a higher threshold. We conclude that the greedy approach may not be able to find a feasible strategy efficiently if most of the strategies in an MDP do not satisfy the specification.

We also assess the effect of adding additional constraints to the task, i.e., surveilling the next green region within a certain time step. We synthesize different policies for different expected times until the UAV reaches the next region. We summarize the results in Table II. For each different probability thresholds (labeled as "β") and expected times to complete the mission (labeled as "κ"), we report the synthesis time and the maximal deviation. The results in Table II illustrate that the maximal deviation δ_{QCP} increases with increasing threshold and decreasing expected time to complete the mission. For example, with the threshold $\beta = 0.9$ and expected time $\kappa = 20$, the maximal deviation between the human and the repaired strategy is 0.598, which shows that the strategies of the human and the autonomy protocol can be significantly different in some states. On the other hand, with the threshold $\beta = 0.7$ and expected time $\kappa = 80$, the maximal deviation between the human strategy and the repaired strategy is 0.054, which is significantly smaller than the previous examples. We also note that there is no significant difference in synthesis time for different thresholds and expected times.

VI. Conclusion and Critique

We introduced a formal approach to synthesize an autonomy protocol in a shared control setting subject to probabilistic temporal logic specifications. The proposed approach utilizes inverse reinforcement learning to compute an abstraction of a human's behavior as a randomized strategy in a Markov decision process. We designed an autonomy protocol such that the resulting robot strategy satisfies safety and performance specifications. We also ensured that the resulting robot behavior is as similar to the behavior induced by the human's commands as possible. We synthesized the robot behavior using quasiconvex programming. We showed the practical usability of our approach through case studies involving autonomous wheelchair navigation and unmanned aerial vehicle planning.

There is a number of limitations and also possible extensions of the proposed approach. First of all, we computed a globally optimal strategy by bisection, which requires checking feasibility of several linear programming problems. A convex formulation of the shared control synthesis problem would make computing the globally optimal strategy more efficient.

We assumed that the human's commands are consistent through the whole execution. This assumption implies the human does not adapt the strategy to the assistance. In the future we will handle non-consistent commands by utilizing additional side information, such as task specifications.

Finally, in order to generalize the proposed approach to other task domains, it is worth to explore transfer learning [23] techniques that allow to handle different scenarios without requiring to relearn the human strategy.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, page 1. ACM, 2004.

[2] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[3] Mihir Bellare and Phillip Rogaway. The complexity of approximating a nonlinear program. In *Complexity in numerical optimization*, pages 16–32. World Scientific, 1993.

[4] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[5] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A storm is coming: A modern probabilistic model checker. In *CAV*, pages 592–600. Springer, 2017.

[6] Anca D. Dragan and Siddhartha S. Srinivasa. Formalizing assistive teleoperation. In *Robotics: Science and Systems*, 2012.

[7] Anca D. Dragan and Siddhartha S. Srinivasa. A policy-blending formalism for shared control. *I. J. Robotic Res.*, 32(7):790–805, 2013.

[8] Andrew Fagg, Michael Rosenstein, Robert Platt, and Roderic Grupen. Extracting user intent in mixed initiative teleoperator control. In *Intelligent Systems Technical Conference*, page 6309, 2004.

[9] Lu Feng, Clemens Wiltsche, Laura Humphrey, and Ufuk Topcu. Synthesis of human-in-the-loop control protocols for autonomous systems. *IEEE Trans. Automation Science and Engineering*.

[10] Vojtěch Forejt, Marta Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. Quantitative multi-objective verification for probabilistic systems. In *TACAS*, pages 112–127. Springer, 2011.

[11] Roland Fryer and Matthew O Jackson. A categorical model of cognition and biased decision making. *The BE Journal of Theoretical Economics*, 8(1).

[12] Jie Fu and Ufuk Topcu. Synthesis of shared autonomy policies with temporal logic specifications. *IEEE Transactions on Automation Science and Engineering*, 13(1):7–17, 2016.

[13] F. Galán, M. Nuttin, E. Lew, P. W. Ferrez, G. Vanacker, J. Philips, and J. del R. Millán. A brain-actuated wheelchair: Asynchronous and non-invasive brain-computer interfaces for continuous control of robots. *Clinical Neurophysiology*, 119(9):2159–2169, 2016/05/28.

[14] Gurobi Optimization, Inc. Gurobi optimizer reference manual. url=http://www.gurobi.com, 2013.

[15] Ernst Moritz Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Omega-regular objectives in model-free reinforcement learning. In *TACAS*, pages 395–412. Springer, 2019.

[16] Nils Jansen, Murat Cubuktepe, and Ufuk Topcu. Synthesis of shared control protocols with provable safety and performance guarantees. In *ACC*, pages 1866–1873. IEEE, 2017.

[17] Shervin Javdani, J Andrew Bagnell, and Siddhartha Srinivasa. Shared autonomy via hindsight optimization. In *Robotics: Science and Systems*, 2015.

[18] Dae-Jin Kim, Rebekah Hazlett-Knudsen, Heather Culver-Godfrey, Greta Rucks, Tara Cunningham, David Portee, John Bricout, Zhao Wang, and Aman Behal. How autonomy impacts performance and satisfaction: Results from a study with spinal cord injured subjects using an assistive robot. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 42(1):2–14, 2012.

[19] Jonathan Kofman, Xianghai Wu, Timothy J Luu, and Siddharth Verma. Teleoperation of a robot manipulator using a vision-based human-robot interface. *IEEE Transactions on industrial electronics*, 52(5):1206–1219, 2005.

[20] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[21] Adam Leeper, Kaijen Hsiao, Matei Ciocarlie, Leila Takayama, and David Gossow. Strategies for human-in-the-loop robotic grasping. In *HRI*, pages 1–8. IEEE, 2012.

[22] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. Siam, 1994.

[23] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[24] Shashank Pathak, Erika Ábrahám, Nils Jansen, Armando Tacchella, and Joost-Pieter Katoen. A greedy approach for the efficient repair of stochastic models. In *NFM*, volume 9058 of *LNCS*, pages 295–309. Springer, 2015.

[25] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[26] Jian Shen, Javier Ibanez-Guzman, Teck Chew Ng, and Boon Seng Chew. A collaborative-shared control system with safe obstacle avoidance capability. In *Robotics, Automation and Mechatronics*, volume 1, pages 119–123. IEEE, 2004.

[27] Brian D Ziebart. Modeling purposeful adaptive behavior with the principle of maximum causal entropy. 2010.

[28] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. 2008.