Mixed-Initiative Control Synthesis: Estimating an Unknown Task Based on Human Control Input ⋆

Sofie Ahlberg * **Dimos V. Dimarogonas** *

* *The Division of Decision and Control Systems, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Sweden.* sofa@kth.se, dimos@kth.se

Abstract: In this paper we consider a mobile platform controlled by two entities; an autonomous agent and a human user. The human aims for the mobile platform to complete a task, which we will denote as the human task, and will impose a control input accordingly, while not being aware of any other tasks the system should or must execute. The autonomous agent will in turn plan its control input taking in consideration all safety requirements which must be met, some task which should be completed as much as possible (denoted as the robot task), as well as what it believes the human task is based on previous human control input. A framework for the autonomous agent and a mixed initiative controller are designed to guarantee the satisfaction of the safety requirements while both the human and robot tasks are violated as little as possible. The framework includes an estimation algorithm of the human task which will improve with each cycle, eventually converging to a task which is similar to the actual human task. Hence, the autonomous agent will eventually be able to find the optimal plan considering all tasks and the human will have no need to interfere again. The process is illustrated with a simulated example.

Keywords: Temporal Logic, Human-Machine Interface, Adaptive Control, Estimation Algorithms, Formal Methods

1. INTRODUCTION

As the use of robots increases it becomes imperative to address issues for human-in-the-loop systems. To construct safe and effective systems we need to consider how the robots should react to the actions of the humans and how the robot best helps the human achieve their goal or adapt in a way that satisfies both entities' desires. Many aspects of this have been considered in previous work. For instance, Carr et al. (2018) and Schlossman et al. (2019) both study the behaviour of humans and use the constructed models to improve the system. In the former, a control policy was created with the aim that the system should act similarly to the human, allowing other humans to be more comfortable around it and for the system to take over simple tasks which humans would otherwise perform. In the latter the constructed system acts like a supervisor handing out tasks to workers, and the human model is used to determine when tasks should be given in order to reduce stress and improve efficiency. In Cao et al. (2010), the roles are reversed and the human is the one acting like a supervisor by assigning tasks to robots in a multi-agent system, in turn she receives feedback from the system in the form of rewards determined by the choices she has made. In Okunev et al. (2012), where both the human and the robot have direct impact on the control input. The issue addressed is to ensure safety regardless of the actions of the human. This is done by applying navigation functions around unsafe areas and a mixed-initiative control policy, guaranteeing that the human won't be able to violate safety. In this paper we will consider a similar setup in the sense that we focus on a single robot which is co-piloted by a human and an autonomous agent. However, in our case, instead of assuming that they are cooperating, we consider that each has their own task to complete. The aim is then to construct a system which guarantees safety while finding a solution which satisfies both tasks as much as possible.

With this in mind, we will use temporal logic to express the tasks. More specifically, we will consider Metric Interval Temporal Logic (MITL), which is built with logic connectors, timed logic operators and boolean valued variables, and allows to express tasks as logic formulas while still being close to English. It has been studied and used in multiple papers such as Zhou et al. (2016), Souza and Prabhakar (2007), Ouaknine and Worrell (2005), Bouyer (2009), Maler et al. (2006) and Brihaye et al. (2013). In the latter is was shown that a MITL formula can be translated into a timed automata Alur and Dill (1994), Alur (1999). This allows to use formal methods for control synthesis to construct a framework for finding a path which satisfies a given MITL task. This has been done for MITL as well as other temporal logic languages in Fainekos et al. (2009), Kloetzer and Belta (2008), Kantaros and Zavlanos (2016) and Fu and Topcu (2015) among others. We suggested a framework for this for a multi-agent system in Andersson

⋆ This work was supported by the H2020 ERC Consolidator Grant LEAFHOUND, the Swedish Foundation for Strategic Research, the Swedish Reasearch Council and the Knut and Alice Wallenberg Foundation.

et al. (2017), which is however not suitable for solving the problem at hand since we have relaxed the goal from finding a completely satisfying path to a maximally satisfying one. This has been addressed in previous literature through formula revision Fainekos (2011), Lahijanian and Kwiatkowska (2016), and approximative simulations Girard and Pappas (2007), Fainekos and Pappas (2009), yet not for the case of MITL. In the line of the latter we introduced the novel metric hybrid distance in Andersson and Dimarogonas (2018) which quantifies how much a MITL formula is violated by a specific trajectory. This was further expanded in Ahlberg and Dimarogonas (2019) to address the concept of hard and soft constraints, allowing us to find a path which guarantees satisfaction of a given hard constraint while minimizing the violation of a given soft constraint with respect to the hybrid distance. This was presented in the setting of a human and robot working together to complete a task. As mentioned above, we have removed here the assumption that the entities are cooperating by considering that the two have their own task to complete. Furthermore, we assume that neither the autonomous agent nor the human has any knowledge of what the other one is attempting to do.

Our approach to the problem at hand is for the autonomous agent to plan for the maximal satisfaction of its own task while attempting to determine what task the human has in mind, allowing it to then plan for that task as well. These tasks are both considered as soft constraints. Simultaneously, the system must consider a safety requirement which is to be handled as a hard constraint. The main contribution of this paper is the framework which the system uses to replan based on newly found knowledge as well as an algorithm for estimating the human task based on human control input. We then use algorithms and solutions from our previous work Andersson and Dimarogonas (2018), Ahlberg and Dimarogonas (2019) to address the mixed-initiative control policy needed to guarantee safety and the planning for finding a least-violating path for hard and soft constraints with respect to the hybrid distance.

2. PRELIMINARIES AND NOTATION

In this paper we will use a weighted transition system as an abstraction of the dynamics of the robot and the environment it is located within.

Definition 1. A *Weighted Transition System* (WTS) is a tuple $T = (\Pi, \Pi_{init}, \rightarrow, AP, L, d)$ where $\Pi = \{\pi_i : i = 0, ..., M\}$ is a finite set of states, $\Pi_{init} \subset \Pi$ is a set of initial states, $\rightarrow \subseteq \Pi \times \Pi$ is a transition relation; the expression $\pi_i \rightarrow \pi_k$ is used to express transition from π_i to π_k, AP is a finite set of atomic propositions, $L : \Pi \rightarrow 2^{AP}$ is an labelling function and $d :\rightarrow\rightarrow \mathbb{R}_+$ is a positive weight assignment map; the expression $d(\pi_i, \pi_k)$ is used to express the weight assigned to the transition $\pi_i \rightarrow \pi_k$.

A discrete path through a WTS with corresponding time stamps is defined as a timed run.

Definition 2. A *timed run* $r^t = (\pi_0, \tau_0)(\pi_1, \tau_1)...$ of a WTS T is an infinite sequence where $\pi_0 \in \Pi_{init}$, $\pi_j \in \Pi$, and $\pi_j \rightarrow \pi_{j+1}$ $\forall j \geq 1$ s.t. $\tau_0 = 0$ and $\tau_{j+1} = \tau_j + d(\pi_j, \pi_{j+1})$, $\forall j \geq 1$.

Table 1. Operators categorized according to the *temporally bounded/non-temporally bounded* notation and Definition 4.

Operator	$b = \infty$	$b \neq \infty$
$\Box_{[a,b]}$	non-temporally bounded, type II	temporally bounded
$\Diamond_{[a,b]}$	non-temporally bounded, type I	temporally bounded
$\mathcal{U}_{[a,b]}$	non-temporally bounded, type I	temporally bounded

The tasks which the robot aims to complete are expressed using metric interval temporal logic:

Definition 3. The *syntax of MITL* over a set of atomic propositions AP is defined by the grammar $\phi := \top \mid ap \mid \neg\, \phi \mid \phi \wedge \psi \mid \phi \mathcal{U}_{[a,b]}\, \psi$ where $ap \in AP$, $a, b \in [0, \infty]$ and ϕ, ψ are formulas over AP. The operators are *Negation* ($\neg$), *Conjunction* ($\wedge$) and *Until* ($\mathcal{U}$) respectively. Given a timed run $r^t = (\pi_0, \tau_0)(\pi_1, \tau_1), ...$ of a WTS, the semantics of the satisfaction relation is then defined as Souza and Prabhakar (2007), Ouaknine and Worrell (2005):

$$(r^t, i) \models ap \Leftrightarrow L(\pi_i) \models ap \text{ (or } ap \in L(\pi_i)), \quad (1a)$$

$$(r^t, i) \models \neg\phi \Leftrightarrow (r^t, i) \not\models \phi, \quad (1b)$$

$$(r^t, i) \models \phi \wedge \psi \Leftrightarrow (r^t, i) \models \phi \text{ and } (r^t, i) \models \psi, \quad (1c)$$

$$(r^t, i) \models \phi\, \mathcal{U}_{[a,b]}\, \psi \Leftrightarrow \exists j \in [a, b],\ s.t.\ (r^t, j) \models \psi \text{ and } \forall i \leq j, (r^t, i) \models \phi. \quad (1d)$$

From this we can define the extended operators *Eventually* ($\Diamond_{[a,b]}\phi = \top \mathcal{U}_{[a,b]} \phi$) and *Always* ($\Box_{[a,b]}\phi = \neg\Diamond_{[a,b]}\neg\phi$).

The operators $\mathcal{U}_I$, $\Diamond_I$ and $\Box_I$, are bounded by the interval $I = [a, b]$, which indicates that the operator should be satisfied within $[a, b]$. We will denote time bounded operators with $b \neq \infty$ as *temporally bounded* operators. All operators that are not included in the set of temporally bounded operators, are called *non-temporally bounded* operators. The operator $\mathcal{U}_I$ can be temporally bounded (if a deadline is associated to the second part of the formula) but contains a non-temporally bounded part. When we use the term *violating non-temporally bounded operators*, we refer to the non-temporally bounded part of an operator being violated. A formula ϕ which contains a temporally bounded operator will be called a temporally bounded formula. The same holds for non-temporally bounded formulas. An MITL specification ϕ can be written as $\phi = \bigwedge_{i \in \{1,2,...,n\}} \phi_i = \phi_1 \wedge \phi_2 \wedge ... \wedge \phi_n$ for some $n > 0$ and some subformulas ϕ_i. In this paper, the notation subformulas ϕ_i of ϕ, refers to the set of subformulas which satisfies $\phi = \bigwedge_{i \in \{1,2,...,n\}} \phi_i$ for the largest possible choice of n such that $\phi_i \neq \phi_j$ $\forall i \neq j$. At every point in time a subformula can be evaluated as satisfied, violated or uncertain. If the subformula is non-temporally bounded there are only two possible outcomes, either uncertain/violated or uncertain/satisfied. We use *Type I* and *Type II* notation:

Definition 4. Andersson and Dimarogonas (2018) A non-temporally bounded formula is denoted as *Type I* if it cannot be concluded to be violated at any time, and as *Type II* if it cannot be concluded to be satisfied at any time. Table 1 shows the categorization.

The *hybrid distance* Andersson and Dimarogonas (2018) is a metric which shows the degree of violation of a

run with respect to a given MITL formula. A plan can violate a formula i) by continuous violation, i.e. exceeding deadlines, or ii) by discrete violation, i.e. the violation of non-temporally bounded operators. We quantify these violations with a metric with respect to time:

Definition 5. The hybrid distance d_h is a satisfaction metric with respect to a MITL formula ϕ and a timed run $r^t = (\pi_0, \tau_0), (\pi_1, \tau_1), ..., (\pi_m, \tau_m)$, defined as: $d_h = hd_c + (1-h)d_d$, where d_c and d_d are the *continuous and discrete distances* between the run and the satisfaction of ϕ, such that $d_c = \sum_{i \in X} T_i^c$, and $d_d = \sum_{j=0,1,...,m} T_j^d$, where X is the set of clocks (given next in Definition 7), T_i^c is the time which the run violates the deadline expressed by clock i, $T_j^d = 0$ if no non-temporally bounded operators are violated by the action $L(\pi_j)$ and $T_j^d = \tau_j - \tau_{j-1}$ otherwise, and $h \in [0, 1]$ is the weight assigning constant which determines the priority between continuous and discrete violations.

To be able to calculate d_h we define its derivative:

Definition 6. $\Phi_H = (\dot{d}_c, \dot{d}_d)$, is a tuple, where $\dot{d}_c \in \{0, ..., n_c\}$ and $\dot{d}_d \in \{0, 1\}$, and $n_c = |X|$ is the number of time bounds associated with the MITL specification.

In Andersson and Dimarogonas (2018), we introduced an extension of the timed Büchi automaton (TBA) Alur and Dill (1994) denoted Timed Automaton with hybrid distance or TAhd for short:

Definition 7. Andersson and Dimarogonas (2018) A *Timed Automaton with hybrid distance* (TAhd) is a tuple $A_H = (S, S_0, AP, X, F, I_X, I_H, E, H, \mathcal{L})$ where $S = \{s_i : i = 0, 1, ...m\}$ is a finite set of locations, $S_0 \subseteq S$ is the set of initial locations, 2^{AP} is the alphabet (i.e. set of actions), where AP is the set of atomic propositions, $X = \{x_i : i = 1, 2, ..., n_c\}$ is a finite set of clocks (n_c is the number of clocks), $F \subseteq S$ is a set of accepting locations, $I_X : S \to \Phi_X$ is a map of clock constraints, $H = (d_c, d_d)$ is the hybrid distance, $I_H : S \to \Phi_H$ is a map of hybrid distance derivative, where I_H is such that $I_H(s) = (d_1, d_2)$ where d_1 is the number of temporally bounded operators violated in s, and $d_2 = 0$ if no non-temporally bounded operators are violated in s and $d_2 = 1$ otherwise, $E \subseteq S \times \Phi_X \times 2^{AP} \times S$ is a set of edges, and $\mathcal{L} : S \to 2^{AP}$ is a labelling function.

The notation $(s, g, a, s') \in E$ is used to state that there exists an edge from s to s' under the action $a \in 2^{AP}$ where the valuation of the clocks satisfy the guard $g = I_X(s) \subseteq \Phi_X$. The expressions $d^c(s)$ and $d^d(s)$ are used to denote the hybrid distance derivatives $\dot{d}_c$ and $\dot{d}_d$ assigned to s by I_H.

Definition 8. Alur and Dill (1994) A *clock constraint* Φ_x is a conjunctive formula of the form $x \bowtie a$, where $\bowtie \in \{<, >, \leq, \geq\}$, x is a clock and a is some non-negative constant. Let Φ_X denote the *set of clock constraints* over the set of clocks X.

We will use the notation of automata timed run for a discrete path through an automaton with corresponding time stamps, indicating at which time evaluations each location in the path is reached.

Definition 9. An *automata timed run* $r^t_{A_H} = (s_0, \tau_0), ..., (s_m, \tau_m)$ of A_H, corresponding to the timed run $r^t = (\pi_0, \tau_0), ..., (\pi_m, \tau_m)$, is a sequence where $s_0 \in S_0$, $s_j \in S$, and $(s_j, g_{j+1}, a_{j+1}, s_{j+1}) \in E \ \forall j \geq 1$ such that i) $\tau_j \models g_j$, $j \geq 1$, and ii) $L(\pi_j) \in \mathcal{L}(s_j)$, $\forall j$.

Definition 10. The continuous violation for the automata timed run $r^t_{A_H} = (s_0, \tau_0), ..., (s_m, \tau_m)$ is $d_c(r^t_{A_H}) = \sum_{i=0,...,m-1} d^c(s_i)(\tau_{i+1} - \tau_i)$, and similarly, the discrete violation for the automata timed run is $d_d(r^t_{A_H}) = \sum_{i=0,...,m-1} d^d(s_i)(\tau_{i+1} - \tau_i)$, and hence the hybrid distance, d_h, as defined in Definition 5, is equivalently given with respect to an automata timed run as

$$d_h(r^t_{A_H}, h) = \sum_{i=0}^{m-1} (hd^c(s_i) + (1-h)d^d(s_i))(\tau_{i+1} - \tau_i) \quad (2)$$

Definition 11. Given a weighted transition system $T = (\Pi, \Pi_{init}, \Sigma, \to, AP, L, d)$ and a timed automaton with hybrid distance $A_H = (S, S_0, AP, X, F, I_X, I_H, E, H, \mathcal{L})$ their *Product Automaton* (P) is defined as $T^p = T \otimes A_H = (Q, Q^{init}, \rightsquigarrow, d, \mathcal{F}, AP, \mathcal{L}^p, I_X^p, I_H^p, X, H)$, where $Q \subseteq \{(\pi, s) \in \Pi \times S : L(\pi) \in \mathcal{L}(s)\} \cup \{(\pi, s) \in \Pi_{init} \times S_0\}$ is the set of states, $Q^{init} = \Pi_{init} \times S_0$ is the set of initial states, $\rightsquigarrow$ is the set of transitions defined such that $q \rightsquigarrow q'$ if and only if i) $q = (\pi, s)$, $q' = (\pi', s') \in Q$, ii) $(\pi, \pi') \in \to$, and iii) $\exists\, g, a$, s.t. $(s, g, a, s') \in E$, $d(q, q') = d(\pi, \pi')$ if $(q, q') \in \rightsquigarrow$, is a positive weight assignment map, $\mathcal{F} = \{(\pi, s) \in Q : s \in F\}$, is the set of accepting states, $\mathcal{L}^p(q) = L(\pi)$ is an observation map, $I_X^p(q) = I_X(s)$ is a map of clock constraints, and $I_H^p(q) = I_H(s)$ is a map of hybrid distance derivative constraints.

3. PROBLEM FORMULATION

The aim in this paper is to design a controller u for the mobile platform such that both the human and the autonomous agent have impact on the resulting trajectory while guaranteeing satisfaction of safety requirements, as well as finding the robotic control input u_r which satisfies both the robot task and the unknown human task as much as possible. From here on, we will denote the MITL formulas expressing the considered tasks as follows; human task as ϕ_h, robot task as ϕ_r, and safety requirements as ϕ_s.

There are then four subproblems which need to be addressed to find a solution to the stated problem; 1) Finding an initial control input u_r such that the closed-loop system satisfies all safety requirements ϕ_s completely and the robot task ϕ_r as much as possible, 2) Designing a control policy such that the human has as much input as possible while guaranteeing satisfaction of all safety requirements ϕ_s, 3) Continuously updating the control input u_r to adapt for the human input u_h, and 4) Estimating the human task u_h and replanning accordingly.

In previous work we have solved the control problem for a system under a hard constraint ϕ^{hard} and a soft constraint ϕ^{soft}. Here, we will use this by identifying ϕ^{hard} and ϕ^{soft} in each subproblem, formally;

Problem 1. Find an initial control policy u_r such that the closed-loop system

$$\dot{x} = Ax + Bu \tag{3}$$

$$u = u_r,\ x(0) = x_0 \tag{4}$$

satisfies the hard constraint $\phi^{hard} = \phi_s$ completely and the soft constraint $\phi^{soft} = \phi_r$ as much as possible, using the hybrid distance as the metric of satisfaction.

Here $\dot{x} = Ax + Bu$ are the of the robot.

Problem 2. Design the mixed-initiative control policy $u = u_r + \kappa u_h$ such that the closed-loop system

$$\dot{x} = Ax + Bu \tag{5}$$

$$u = u_r + \kappa u_h,\ x(0) = x_0,\ x \in X \tag{6}$$

satisfies the hard constraint $\phi^{hard} = \phi_s$ completely while allowing the human as much control u_h as possible. Here, $\kappa \in X \times \phi^{hard} \to [0, 1]$ is a mapping from position and hard task onto a weight constant between 0 and 1.

Problem 3. Continuosly update u_r to maximize the satisfaction of the soft constraint $\phi^{soft} = \phi_r \wedge \phi_h^{est}$, while guaranteeing satisfaction of the hard constraint $\phi^{hard} = \phi_s$, to adapt for the differences between the planned trajectory following u_r and the one resulting from following u. Here ϕ_h^{est} refers to the estimation of ϕ_h and is initially empty.

Problem 4. Estimate the human specification ϕ_h, as ϕ_h^{est}, based on previous human control input u_h, and redesign the control policy u_r s.t. the closed-loop system

$$\dot{x} = Ax + Bu \tag{7}$$

$$u = u_r + \kappa u_h,\ x(0) = x_0 \tag{8}$$

satisfies the hard constraint $\phi^{hard} = \phi_s$ completely, and satisfies the soft constraint $\phi^{soft} = \phi_r \wedge \phi_h^{est}$ as much as possible, using the hybrid distance as the metric of satisfaction. We assume that $\phi_h = \bigwedge_{i=1}^{k} \Diamond_{I_i} a_i$ where $I_i = [0, t_i]$, i.e. that the human task consists of visiting a finite set of areas a_i in the workspace within some deadlines.

4. CONTROL DESIGN

4.1 Initial Robotic Control

Subproblem 1 was addressed in Ahlberg and Dimarogonas (2019). Here we will give a brief overview. For a given hard specification (safety) ϕ_s and a given soft specification (robot task) ϕ_r, a control policy u_r, which satisfies ϕ_s and minimizes the violation of ϕ_r, can be found by the following steps;

i) Abstract the environment and dynamics of the mobile platform into a weighted transition system $T = (\Pi, \Pi_{init}, \to, AP, L, d)$ where the weights d corresponds to the worst case transition times, as described in Andersson et al. (2017).
ii) Construct a Timed Automaton with hybrid distance $A_H = (S, S_0, AP, F, I_X, I_H, E, H, \mathcal{L})$ from the specifications ϕ_s and ϕ_r as described in Ahlberg and Dimarogonas (2019).
iii) Construct the product $P = (Q, Q^{init}, \rightsquigarrow, d, \mathcal{F}, AP, \mathcal{L}^p, I_X^p, I_H^p, X, H)$ of T and A_H.
iv) Use a graph search algorithm, such as the modified Dijkstra algorithm suggested in Andersson and Dimarogonas (2018), to find the control input u_r which minimizes the hybrid distance for a given value of h. The suggested algorithm for this is Alg. 1 where $d_c^0 = d_d^0 = d_h^0 = 0$. That is, we set the initial values of all distances to 0 and search for the shortest path between the initial state and an accepting state using d_h as the distance.

Algorithm 1. *dijkstraHD*()
% Dijkstra Algorithm with Hybrid Distance as cost function
Data: P, h, d_c^0, d_d^0, d_h^0
Result: r_{hd}^{min}, d_h, d_c, d_d
Q =*set of states;* q_0 =*initial state;* $SearchSet = q_0$; $d(q, q')$ =*weight of transition* $q \rightsquigarrow q'$ *in* P
if $q = q_0$ **then**
  $d_h(q) = d_h^0$, $d_c(q) = d_c^0$, $d_d(q) = d_d^0$
else
  $d_h(q) = d_c(q) = d_d(q) = \infty$
end
for $q \in Q$ **do**
  $pred(q) = \emptyset$
end
while *no path found* **do**
  Pick $q \in SearchSet$ *s.t.* $q = \arg\min(d_h(q))$
  if $q \in F$ **then**
    path found
  end
  else
    find all q' *s.t.* $q \rightsquigarrow q'$
    for *every* q' **do**
      $d_h^{step} = (h\dot{d_c}(q) + (1-h)\dot{d_d}(q))d(q, q')$
      if $d_h(q') > d_h(q) + d_h^{step}$ **then**
        update $d_h(q')$, $d_c(q')$, $d_d(q')$ *and* $pred(q')$ *and add* q' *to* $SearchSet$ *Remove* q *from* $SearchSet$
      end
    end
  end
end
while $q \neq q_0$ **do**
  use $pred(q)$ to iteratively form the path back to q_0 $\to r_{hd}^{min}$
end

Applying the resulting control input without human input i.e. $u = u_r$ will correspond to the high level plan which violates the ϕ_r the least, while satisfying ϕ_s completely.

4.2 Mixed-Initiative Control

Subproblem 2, i.e. designing the mixed-initiative controller to ensure safety, was also addressed in Ahlberg and Dimarogonas (2019). The idea is to give the human as much influence as possible at all time, and only restricting her to not enter states which would violate ϕ_s. This is done by constructing a set of pairs of states and time-stamps $Q_T^t = \{(q, t) : q \in Q_T, t \geq 0\}$ where Q_T is the set of states from which accepting states $\mathcal{F}$ are not reachable, and t are time-stamps corresponding to the minimum time required to enter q. We can then design κ s.t.

$$\kappa \in \begin{cases} 0 & \text{if } d_t < d_s \\ (0, 1) & \text{if } d_t \in (d_s, d_s + \varepsilon) \\ 1 & \text{if } d_t > d_s + \varepsilon \end{cases} \tag{9}$$

where $d_s > 0$ and $\varepsilon > 0$ are design parameters and d_t is defined as $d_t = \min_{(q,t)\in Q_T^t} dist(x,(q,t))$ for

$$dist(x,(q,t)) = \begin{cases} \|x - proj(q,T)\| & \text{if } t_0 + d(\pi_0, \\ & proj(q,T)) > t \\ \infty & \text{otherwise,} \end{cases} \quad (10)$$

where π_0 and t_0 corresponds to the current location and the valuation of time and the projection of a state onto the transition system (and onto the automaton) are defined as:

Definition 12. The projections of a timed run of a product automaton $r_P^t = (\pi_1, s_1)(\pi_2, s_2), ..., (\pi_m, s_m)$ onto a TAhd A_H and a WTS T are defined as $proj(r_P^t, A_H) = s_1, s_2, ..., s_m$ and $proj(r_P^t, T) = \pi_1, \pi_2, ..., \pi_m$.

That is, d_t is the minimum distance to any state in Q_T^t which can be reached in the given transition time. This is satisfied by

$$\kappa(x, Q_T^t) = \frac{\rho(d_t - d_s)}{\rho(d_t - d_s) + \rho(\varepsilon + d_s - d_t)} \quad (11)$$

where $\rho(s) = e^{-1/s}$ for $s > 0$ and $\rho(s) = 0$ for $s \leq 0$ which will take on the values:

$$\kappa = \begin{cases} 0 & \text{if } d_t < d_s \\ \frac{e^{1/(d_s - d_t)}}{e^{1/(d_s-d_t)} + e^{1/(d_t - d_s - \varepsilon)}} \in (0,1) & \text{if } d_t \in (d_s, d_s + \varepsilon) \\ 1 & \text{if } d_t > d_s + \varepsilon \end{cases} \quad (12)$$

4.3 Updating Robotic Control Policy

In this section we address subproblem 3, i.e. updating the robotic control policy u_r to minimize the violation of ϕ^{soft} given the actions of the human. Initially we will use $\phi^{soft} = \phi_r$, this will then be updated to $\phi^{soft} = \phi_r \wedge \phi_h^{est}$. The approach is to re-run Alg. 1 on the product automaton P (constrcuted from the TAhd representing $\phi^{soft} \wedge \phi^{hard}$) where the initial state is set as the current state considering progress made in A_H and current state in T. When re-running the search algorithm the setting of the initial values of the distances (hybrid d_h^0, discrete d_d^0 and continuous d_c^0) should also be updated to the current valuations. The initial state and distances are found by considering the trajectory which has been followed by the mobile platform so far. More specifically, if the result of $u = u_r + \kappa u_h$ has been the trajectory which corresponds to the discrete path $(\pi_0, t_0), (\pi_1, t_1), (\pi_2, t_2), ..., (\pi_m, t_m)$ and the automata run $r_{A_H}^t = (s_0, t_0'), (s_1, t_1'), (s_2, t_2'), ..., (s_i, t_i')$, then $Q^{init} = (\pi_m, s_i)$, $d_c^0 = d_c(r_{A_H}^t)$, $d_d^0 = d_d(r_{A_H}^t)$ and $d_h^0 = d_h(r_{A_H}^t, h)$ (from definition 10).

4.4 Estimating Human Task and Re-planning

Finally, we consider subproblem 4, i.e. estimating ϕ_h to find the optimal plan given the specifications known by the robot. During a run, the human will have maximal control as long as she is not violating a safety constraint. She will stop interfering (i.e. $u_h = 0$) when ϕ_h is completed as well as possible. Any region in the resulting discrete path is potentially a goal in the human task. Here, we use the term goal to denote a label of a region which the human task includes visiting. The last region which the human actively steers the robot into, π_h^{last}, must then be a goal since the human task wasn't satisfied prior to arriving in the region but was so afterwards. We can therefore conclude that the label $L(\pi_h^{last})$ is a goal in ϕ_h. We can then construct our first estimate of ϕ_h such that $\phi_h^{est} = \Diamond_I L(\pi_h^{last})$ where $I = [0, T]$, where T is the time of arrival at π_h^{last} in the resulting trajectory. This is depicted in Alg. 2. The robot can then replan to find u_r by planning for the task $\phi_s \wedge \phi_r \wedge \phi_h^{est}$ following the steps in 4.1 (reconstructing the automata, product and re-running the graph-search algorithm).

Algorithm 2. *estimateHumanTask()*
%Algorithm for improving the estimate of ϕ_h
Data: human control input $u_h(t)$, resulting timed run $r^t = (\pi_0, t_0), (\pi_1, t_1), ..., (\pi_m, t_m)$, previous estimate of human task $\phi_h^{est,old}$
Result: $\phi_h^{est,new}$
$T = \max t$ s.t. $u_h(t) \neq 0$
$\pi_h^{last} = \pi \in \Pi$ s.t. $(\pi, t_i) \in r^t$ and $T \in (t_i, t_{i+1})$
$\phi_h^{est,new} = \phi_h^{est,old} \wedge \Diamond_{[0,t_i]} L(\pi_h^{est})$

Here we assume that the human will only interfere to improve the path with respect to her task. Hence, if the human is shown the new plan, she should not try to guide us towards the goals which are part of the plan. By repeating the process until the human no longer interferes, i.e., until $u_h = 0\ \forall t$, the remainder of the goals in the human task can then be added to the estimate. When $u_h = 0$ is constant throughout a run it then follows that ϕ_h^{est} is either identical or similar to ϕ_h. Here we say that ϕ_h^{est} is similar to ϕ_h if all goals in ϕ_h are included in ϕ_h^{est} with potentially inaccurate time intervals I, or any missing goal in ϕ_h^{est} is visited in the resulting discrete plan despite not being planned for. If the human wishes to add further tasks this is possible by guiding the robot to new regions. These will then eventually be added. As a result any goals in ϕ_h which hadn't been added to ϕ_h^{est} may be added in following runs as the need arises if the robot changes its plan to not include them. It is however not possible to remove tasks by simple control action since the human can't indicate that a region doesn't need to be visited by applying control input.

Completion It follows from Alg. 2, that the robot will require at most $k+1$ runs to construct a plan which takes k human goals into consideration. This is due to the fact that one goal is being added at each run if $u_h(t) \neq 0\ \forall t$ and that the robot is able to plan for each added goal during the following run. Hence, the estimation ϕ_h^{est} will converge to a task similar to ϕ_h in at most k runs.

5. EXAMPLE

Consider a workspace in Fig. 1 consisting of a 7 by 6 grid. It requires 1 time unit to move between any two regions in the workspace. The robot starts in region 3 (marked as initial). The goal of the human is to complete the task: $\phi_h = \Diamond_{\leq 3} g_1 \wedge \Diamond_{\leq 5} \wedge g_2 \Diamond_{\leq 15} \wedge g_3 \Diamond_{\leq 20} g_4$, i.e., visiting all green areas. The robot task is: $\phi_r = \Diamond_{\leq 12} b_1 \wedge \Diamond_{\leq 4} g_2 \wedge \neg g_2 \mathcal{U} b_1$, i.e., visiting the blue regions and the light green region, enforcing the order to visit the dark blue before the green. The safety requirement is: $\phi_s = \Box \neg r_1 \wedge \Box \neg r_2$, i.e., avoiding the red areas.

Following the suggested framework the result will be:

(1) First run:
   (a) The autonomous agent plans for: $\phi = \phi_s \wedge \phi_r$ resulting in the path illustrated in Figure 1a, i.e., $3, 10, 17, 18, 19, 26, 25, 24, 23, 22, 29, 36$.
   (b) The human interupts the run and steers the robot into the green regions along the way, while the robot reacts and replans to continue with its own task. The resulting path is illustrated in Figure 2a, i.e., $3, 10, 9, 16, 17, 18, 19, 26, 27, 28, 35, 42, 41, 40, 39, 38, 31, 30, 29, 36$.
   (c) The autonomous agent estimates the human task to be: $\phi_h^{est} = \Diamond_{\leq 16} g_3$, where g_3 was the last region entered due to human input and 16 is the time it was entered upon.

(2) Second run:
   (a) The autonomous agent plans for: $\phi = \phi_s \wedge \phi_r \wedge \phi_h^{est}$ resulting in the path illustrated in Figure 1b, i.e., $3, 10, 17, 18, 19, 26, 25, 24, 31, 38, 37, 36$.
   (b) The human interrupts the run and steers the robot into the remaining green regions along the way, while the robot reacts and replans to continue with its own task. The resulting path is illustrated in Figure 2b, i.e., $3, 10, 9, 16, 17, 18, 19, 26, 25, 24, 31, 38, 39, 40, 41, 42, 41, 40, 39, 38, 37, 36$.
   (c) The autonomous agent estimates the human task to be: $\phi_h^{est} = \Diamond_{\leq 16} g_3 \wedge \Diamond_{\leq 15} g_4$, where g_4 was the last region entered due to human input and 15 is the time it was entered upon.

(3) Third run:
   (a) The autonomous agent plans for: $\phi = \phi_s \wedge \phi_r \wedge \phi_h^{est}$ resulting in the path illustrated in Figure 1c, i.e., $3, 10, 17, 18, 19, 26, 25, 24, 31, 30, 29, 36, 37, 38, 39, 40, 41, 42$.
   (b) The human interrupts the run and steers the robot into the remaining green region, while the robot reacts and replan to continue with its own task. The resulting path is illustrated in Figure 2c, i.e. $3, 10, 9, 16, 17, 18, 19, 26, 25, 24, 31, 30, 29, 36, 37, 38, 39, 40, 41, 42$.
   (c) The autonomous agent estimate the human task to be: $\phi_h^{est} = \Diamond_{\leq 16} g_3 \wedge \Diamond_{\leq 15} g_4 \wedge \Diamond_{\leq 2} g_1$, where g_1 was the last region entered due to human input and 2 is the time it was entered upon.

(4) Fourth run:
   (a) The autonomous agent plans for: $\phi = \phi_s \wedge \phi_r \wedge \phi_h^{est}$ resulting in the same path as the last step in the previous run.
   (b) The humans task is completed and hence she gives no new control input.

It should be noted that the last plan from the autonomous agent doesn't have to be identical to the last path suggested by the human. If the human has chosen a non-optimal path which leads to greater hybrid distance the system will find the better option. The resulting path meets all criteria since the safety requirement is satisfied and both the human's and the robot tasks are considered. The final estimation of the human task is $\phi_h^{est} = \Diamond_{\leq 16} g_3 \wedge \Diamond_{\leq 15} g_4 \wedge \Diamond_{\leq 2} g_1$. Comparing it to the original task we note two differences; one goal region (g_2) is missing and the times associated with reaching the goal regions are not the same. The first is due to the goal region being part of the path despite not being added to the estimation. In this case, since it was also part of the robot task, this will always be true. The time differences are due to the actions of the human. In two cases (g_1 and g_4) the human managed to steer the mobile platform to the goal regions quicker than required, resulting in a smaller deadline for the estimated task. In the last case (g_3) the human was slower than the original deadline resulting in a more generous time limit for the estimation.

6. CONCLUSIONS AND FUTURE WORK

We have suggested a framework where a human and an autonomous agent co-pilot a mobile platform in order to complete some given tasks. It is assumed that each entity has its own task to perform and that neither has any knowledge of the other's task. The aim is for the autonomous agent to estimate the human task based on human control input, and to then find the path that maximizes the satisfaction of both tasks. A mixed-initiative control policy is applied to ensure that safety requirements are met. The process is illustrated with an example.

For future work the system should be validated with real-time experiments. Other directions to investigate include improved methods to remove goal regions, extending the estimation algorithm to include different forms of specifications and improving the efficiency to add multiple goal regions to the estimation during a single run.

REFERENCES

Ahlberg, S. and Dimarogonas, D.V. (2019). Human-in-the-loop control synthesis for multi-agent systems under hard and soft metric interval temporal logic specifications. *IEEE International Conference on Automation Science and Engineering (CASE)*.

Alur, R. (1999). Timed automata. In *Computer Aided Verification*, 8–22. Springer.

Alur, R. and Dill, D.L. (1994). A theory of timed automata. *Theoretical computer science*, 126(2), 183–235.

Andersson, S. and Dimarogonas, D.V. (2018). Human in the loop least violating robot control synthesis under metric interval temporal logic specifications. *European Control Conference (ECC)*.

Andersson, S., Nikou, A., and Dimarogonas, D.V. (2017). Control synthesis for multi-agent systems under metric interval temporal logic specifications. *20th World Congress of the International Federation of Automatic Control (IFAC WC 2017)*.

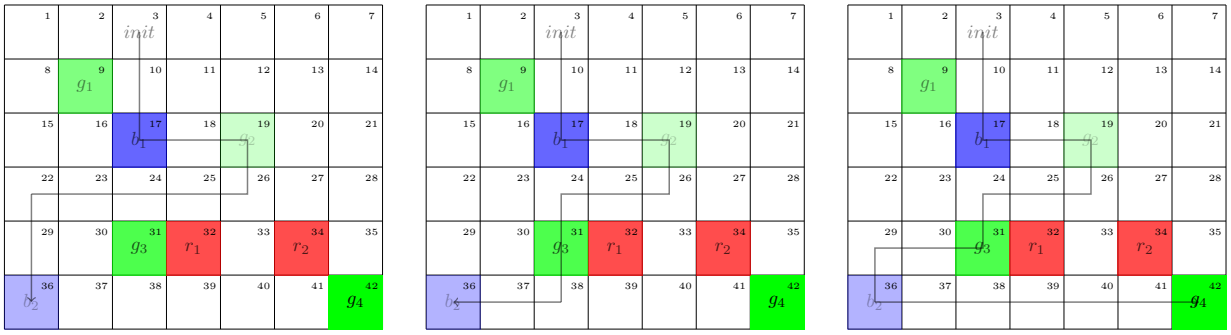
Bouyer, P. (2009). From qualitative to quantitative analysis of timed systems. *Mémoire dhabilitation, Université Paris*, 7, 135–175.

Brihaye, T., Estiévenart, M., and Geeraerts, G. (2013). On mitl and alternating timed automata. In *Formal Modeling and Analysis of Timed Systems*, 47–61. Springer.

Cao, M., Stewart, A., and Leonard, N.E. (2010). Convergence in human decision-making dynamics. *Systems & Control Letters*, 59(2), 87–97.

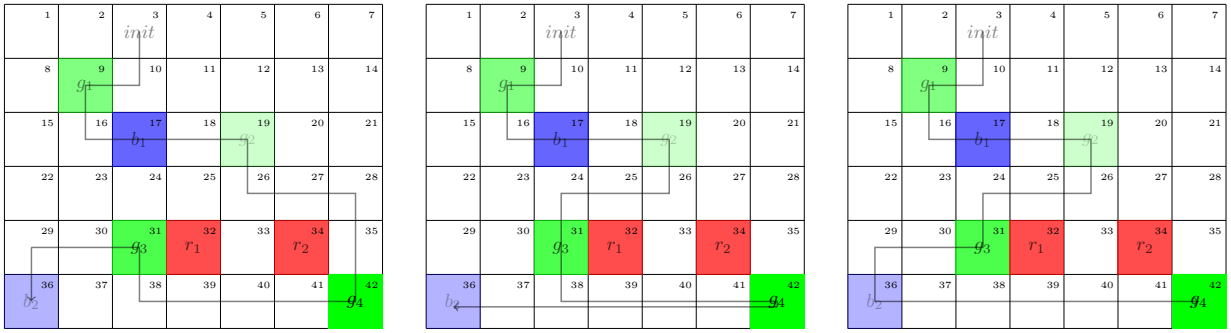
Carr, S., Jansen, N., Wimmer, R., Fu, J., and Topcu, U. (2018). Human-in-the-loop synthesis for partially observable markov decision processes. In *2018 Annual American Control Conference (ACC)*, 762–769. doi: 10.23919/ACC.2018.8431911.

Fainekos, G.E. (2011). Revising temporal logic specifications for motion planning. In *Robotics and Automation*



(a) First run. (b) Second run. (c) Third run.

Fig. 1. Paths suggested by autonomous agent.



(a) First run. (b) Second run. (c) Third run.

Fig. 2. Paths resulting from human input during runs.

(ICRA), 2011 IEEE International Conference on, 40–45. IEEE.

Fainekos, G.E., Girard, A., Kress-Gazit, H., and Pappas, G.J. (2009). Temporal logic motion planning for dynamic robots. *Automatica*, 45(2), 343 – 352. doi: https://doi.org/10.1016/j.automatica.2008.08.008.

Fainekos, G.E. and Pappas, G.J. (2009). Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42), 4262–4291.

Fu, J. and Topcu, U. (2015). Computational methods for stochastic control with metric interval temporal logic specifications. In *2015 54th IEEE Conference on Decision and Control (CDC)*, 7440–7447. IEEE.

Girard, A. and Pappas, G.J. (2007). Approximation metrics for discrete and continuous systems. *IEEE Transactions on Automatic Control*, 52(5), 782–798.

Kantaros, Y. and Zavlanos, M. (2016). A Distributed LTL-Based Approach for Intermittent Communication in Mobile Robot Networks. *American Control Conference (ACC), 2016*, 5557–5562.

Kloetzer, M. and Belta, C. (2008). A fully automated framework for control of linear systems from temporal logic specifications. *Automatic Control, IEEE Transactions on*, 53(1), 287–297.

Lahijanian, M. and Kwiatkowska, M. (2016). Specification revision for markov decision processes with optimal trade-off. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, 7411–7418. IEEE.

Maler, O., Nickovic, D., and Pnueli, A. (2006). From mitl to timed automata. In *Formal Modeling and Analysis of Timed Systems*, 274–289. Springer.

Okunev, V., Nierhoff, T., and Hirche, S. (2012). Human-preference-based control design: Adaptive robot admittance control for physical human-robot interaction. In *2012 IEEE RO-MAN: The 21st IEEE International Symposium on Robot and Human Interactive Communication*, 443–448. doi:10.1109/ROMAN.2012.6343792.

Ouaknine, J. and Worrell, J. (2005). On the decidability of metric temporal logic. In *Logic in Computer Science, 2005. LICS 2005. Proceedings. 20th Annual IEEE Symposium on*, 188–197. IEEE.

Schlossman, R., Kim, M., Topcu, U., and Sentis, L. (2019). Toward achieving formal guarantees for human-aware controllers in human-robot interactions. *arXiv preprint arXiv:1903.01350*.

Souza, D. and Prabhakar, P. (2007). On the expressiveness of mtl in the pointwise and continuous semantics. *International Journal on Software Tools for Technology Transfer*, 9(1), 1–4.

Zhou, Y., Maity, D., and Baras, J.S. (2016). Timed Automata Approach for Motion Planning Using Metric Interval Temporal Logic. *European Control Conference (ECC 2016)*.