Human-in-the-Loop Control Synthesis for Multi-Agent Systems under Hard and Soft Metric Interval Temporal Logic Specifications*

Sofie Ahlberg[1] and Dimos V. Dimarogonas[1]

***Abstract*— In this paper we present a control synthesis framework for a multi-agent system under hard and soft constraints, which performs online re-planning to achieve collision avoidance and execution of the optimal path with respect to some human preference considering the type of the violation of the soft constraints. The human preference is indicated by a mixed initiative controller and the resulting change of trajectory is used by an inverse reinforcement learning based algorithm to improve the path which the affected agent tries to follow. A case study is presented to validate the result.**

I. INTRODUCTION

With the progress in the robotics and autonomous control fields we see an increase in robotic presence in environments populated by humans. This has increased the importance of human robot interaction (HRI) and Human-in-the-Loop planning and control. These include both physical interaction and communication, where it is important to create systems that are safe and receptive to human preference. To achieve safety, we need system designs with guarantees, such as those that can be achieved by formal methods, that aim at control synthesis from high level temporal logic specifications [1], [2], [3]. In this paper, we consider Metric Interval Temporal Logic (MITL) [4],[5], which can be represented by a timed automaton [6]. Our goal is to design a system that is safe, but also adaptive towards human input and the environment. To achieve this the standard control synthesis framework [7] should be extended to handle the case when a desired specification isn't completely satisfiable.

Different approaches have been suggested for solving this problem. In [8] a method for abstraction refinement to find control policies which could not be found in a sparser partitioning was suggested. In [9] a framework which gives feedback on why the specification is not satisfiable and how to modify it was presented. [10] instead treat the environment as stochastic and designs the controller such that the probability of satisfaction is maximized. Here, we will use the metric hybrid distance which we introduced in [11], to find the controller which minimizes the violation. We will also consider specifications consisting of hard and soft constraints, where the hard constraints must be satisfied.

To achieve adaptability towards the humans preference the system must attain the knowledge of what the human priorities and what consequences this should have on its behaviour. This was discussed in [12], where a control policy was created based on data of human decisions. In [13], a model of human workload information was used to optimize the systems behaviour to balance risk of stress due to full backlogs against risk of low productivity due to empty backlogs. Here, we will instead consider humans giving input to the controllers directly through the so-called mixed-initiative control [14]. The idea is to allow human input while still keeping the guarantees of safety which we acquired from the formal method-based synthesis. The same approach was used in [15] but without the added time constraints inherent to MITL. Here, the human preference is considered to be limited to in what manner the soft constraints should be violated, namely if time (deadlines) is higher prioritized than performing non-desired actions (entering states which should preferably be avoided) or vice versa. To convert the human control input into the desired knowledge we will use an inverse reinforcement learning (IRL)[16] approach.

II. PRELIMINARIES AND NOTATION

In this paper, we consider a multi-agent system where each agent has controllable linear dynamics which can be abstracted into a Weighted Transition System (WTS) where the weights are the corresponding transition times.

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 $\pi_i \rightarrow \pi_k$.

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 $\tau_{j+1} = \tau_j + d(\pi_j, \pi_{j+1})$, $\forall j \geq 1$.

MITL is used to express the considered specifications.

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

*This work was supported by the H2020 ERC Starting Grand BUCOPHSYS, the Swedish Foundation for Strategic Research, the Swedish Reasearch Council and the Knut and Alice Wallenberg Foundation.

[1]Sofie Ahlberg and Dimos V. Dimarogonas are with 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`

TABLE I: 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

of the satisfaction relation is then defined as [5], [4]:

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

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

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

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

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 $\phi = \bigwedge_{i\in\{1,2,\ldots,n\}} \phi_i = \phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_n$ 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 $\phi = \bigwedge_{i\in\{1,2,\ldots,n\}} \phi_i$ 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: [11] 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 I shows the categorization.

The *hybrid distance* is a metric which shows the degree of violation of a run with respect to a given MITL formula. It was first introduced in [11] and will be used to find a least violating run with respect to some soft constraints. A plan can violate a MITL formula in two ways; 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 $r^t = (\pi_0, \tau_0), (\pi_1, \tau_1), \ldots, (\pi_m, \tau_m)$, 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 $d_d = \sum_{j=0,1,\ldots,m} T_j^d$, 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 $\dot{d}_c \in \{0, \ldots, n_c\}$ and $\dot{d}_d \in \{0, 1\}$, and $n_c = |X|$ is the number of time bounds associated with the MITL specification.

In [11], we introduced an extension of the timed Büchi automaton (TBA) [17] denoted Timed Automaton with hybrid distance or TAhd for short:

Definition 7: [11] 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 $S = \{s_i : i = 0, 1, \ldots m\}$ 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, $X = \{x_i : i = 1, 2, \ldots, n_c\}$ 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.

The clock constraints are defined as:

Definition 8: [17] 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.

Definition 9: An *automata timed run* $r^t_{A_H} = (s_0, \tau_0), \ldots, (s_m, \tau_m)$ of A_H, corresponding to the timed run $r^t = (\pi_0, \tau_0), \ldots, (\pi_m, \tau_m)$, 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) $L(\pi_j) \in \mathcal{L}(s_j)$, $\forall j$.

It follows from Definitions 7 and 9, that the continuous violation for the automata timed run is $d_c = \sum_{i=0,\ldots,m-1} d^c(s_i)(\tau_{i+1} - \tau_i)$, and similarly, the discrete violation for the automata timed run is $d_d = \sum_{i=0,\ldots,m-1} d^d(s_i)(\tau_{i+1} - \tau_i)$, and hence the hybrid distance, d_h, as defined in Definition 5, is equivalently given with respect to *an automata timed run* as

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

The product of a WTS and a TAhd was presented in [11]:

Definition 10: 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) $(\pi, \pi') \in \rightarrow$, 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.

III. Problem Formulation

The problem considered in this paper is to, for each agent in a multi-agent system, i) find the plan which minimizes the violation of the soft constraint, while satisfying the hard constraint, ii) learn the human preference concerning the type of violation based on human control input, and iii) avoid collisions with other agents via online re-planning. The input of each agent is assumed to be bounded with $|u_i| \leq u_{max},\ \forall i \in \{1, ..., N\}$. The hybrid distance ($d_h$) is used as the measurement of violation, where $d_h = 0$ corresponds to complete satisfaction. The human preference is indicated by the value of h. This can be expressed as four sub-problems:

Problem 1: **Initial plan:** Given a WTS T and an MITL specification $\phi = \phi^{hard} \wedge \phi^{soft}$, find the timed run r^t of T that corresponds to the automata timed run $\hat{r}^t_{A_H}$ that satisfies: $\hat{r}^t_{A_H} = \arg\min_{r^t_{A_H}} d_h(r^t_{A_H}, h)$, where A_H is the TAhd that corresponds to ϕ and $h = 0.5$. That is, find the control policy which guarantees the satisfaction of ϕ^{hard}, and maximizes the satisfaction of ϕ^{soft}, given the preference h.

Problem 2: **Learning preference and updating plan:** Given a human control input u_h, update the estimation of h such that the resulting trajectory (up until this point in time) is optimal with respect to the hybrid distance. Given the updated value of h, find a new plan r^t_{new} (for the remainder of the task) such that $d_h(r^t_{A_H}, h)$ is minimized by the corresponding automata timed run $\overline{r}^t_{A_H}$. Assuming that the human has a value of h in mind and acts accordingly, the updated solution should thus satisfy $d_h(\overline{r}^t_{A_H}, h) < d_h(\hat{r}^t_{A_H}, h)$.

Problem 3: **Collision avoidance:** Given the location of all other agents in the system, find a new plan which doesn't include occupied states and otherwise follows the preferences of the human, if the imminent part of the trajectory crosses the location of another agent.

Problem 4: **Safety:** Design a control law such that the input from the human (u_h) can not cause the agent to violate the hard constraint.

IV. Offline Synthesis of Initial Plan

The solution to Problem 1 is performed offline and follows the outline we suggested in [11]. The framework is decentralized and inspired by the standard 3 steps procedure for single agent control synthesis, i.e. for each agent the planning follows the steps: 1) Construct a Timed Automaton with Hybrid Distance ($TAhd$) which represents the MITL specification. 2) Construct a Product Automaton of the $TAhd$ and a WTS representing the system dynamics. 3) Find the least violating path, i.e, the shortest path with respect to the hybrid distance, d_h, and for $h = 0.5$. The difference between the solution suggested here and the one presented in [11] occurs in the step 1, where we now consider hard constraints as well as soft which alters the construction of the TAhd.

We start with the construction of the TAhd by considering the locations. To describe the construction of locations, we first introduce the evaluation sets φ:

Definition 11: An evaluation set φ_i of a subformula ϕ_i contains the possible evaluations of the subformula: $\varphi_i = \{\phi_i^{unc}, \phi_i^{vio}, \phi_i^{sat}\}$ if ϕ_i is temporally bounded, $\varphi_i = \{\phi_i^{unc}, \phi_i^{sat}\}$ if ϕ_i is non-temporally bounded type I and $\varphi_i = \{\phi_i^{unc}, \phi_i^{vio}\}$ otherwise.

Next we introduce subformula evaluations ψ:

Definition 12: A subformula evaluation ψ of a formula ϕ is one possible outcome of the formula, i.e. a conjunction of elements of the evaluation sets: $\psi = \bigwedge_i \phi_i^{state}$, $\phi_i^{state} \in \varphi_i$. We will use Ψ to denote the set of all subformula evaluations ψ of a formula ϕ, i.e. all possible outcomes of ϕ at any time.

We can now construct the location set $S = \{s_i : i = 1, ..., |\Psi|\}$. Then $S_0 = s_j$ where $\psi_j = \bigwedge_i \phi_i^{unc}$, and $F = s_k$ where $\psi_k = \bigwedge_{i \in I} \phi_i^{sat} \wedge \bigwedge_{j \in J} \phi_j^{unc}$, where $I \cap J = 1, .., |\Psi|$ and J contains the indexes of all ϕ_j which are non-temporally bounded type II (i.e. cannot be evaluated as satisfied). The set of clocks X must include at least one clock for each temporally bounded ϕ_i, two if there is both a lower and an upper bound. I_X is easily constructed such that $s \rightarrow \Phi_X \in I_X$ if $\phi_i^{vio} \notin \psi$ where ϕ_i is temporally bounded by Φ_X. The hybrid distance derivative mapping $I_H(s) = (d_1, d_2)$ is constructed such that d_1 is equal to the number of clock constraints associated with the subformulas $\phi_i^{vio} \in \psi$, and $d_2 = 1$ if any non-temporally bounded subformula $\phi_i^{vio} \in \psi$ and $d_2 = 0$ otherwise. To construct the edges we first introduce some new definitions and notation:

Definition 13: The distance set of two subformula evaluations ψ and ψ' is defined as $|\psi - \psi'| = \{\phi_i : \phi_i^{state'} \neq \phi_i^{state}\}$. That is, it consists of all subformulas ϕ_i which are evaluated differently in the subformula evaluations.

We use $(\psi, g, a) \rightarrow \psi'$ to denote that all subformulas $\phi_i \in |\psi - \psi'|$ are i) uncertain in ψ and ii) it holds that ϕ_i is re-evaluated to $\phi_i^{state'} \in \psi'$ if action a occurs at time t satisfying guard g.

The edges can now be constructed in 4 steps:

i) Construct all edges corresponding to progress such that: $(s, g, a, s') \in E$ if $(\psi, g, a) \rightarrow \psi'$.

ii) Construct edges which correspond to non-temporally bounded soft constraint/s no longer being violated such that $(s, g, a, s') \in E$ if; i) $\forall \phi_i \in |\psi - \psi'|$, $\phi_i \in \phi^{soft}$ and is non-temporally bounded, and $\phi_i^{vio} \in \psi$, and ii) $(s'', g, a, s') \in E$ for some s'' where $|\psi - \psi'| = |\psi - \psi''|$, or i) $\forall \phi_i \in |\psi - \psi'|$, $\phi_i \in \phi^{soft}$ and is non-temporally bounded, and $\phi_i^{vio} \in \psi$, and ii) $(s', g, a', s) \in E$, where $a' = 2^{AP} \backslash a$.

iii) Construct edges which correspond to temporally-bounded soft constraint/s no longer being violated such that $(s, g, a, s') \in E$ if; i) $\exists \phi_i \in |\psi - \psi'|$, such that $\phi_i \in \phi^{soft}$ is temporally bounded, and $\phi_i^{vio} \in \psi$, $\phi_i^{sat} \in \psi'$, $\phi_i^{unc} \in \psi''$, where $(s'', g', a, s') \in E$, and $(s'', g, a, s) \in E$, ii) $g = g' \backslash \Phi_{X_i}$, where X_i is the set of clocks associated with ϕ_i s.t. $\phi_i^{unc} \in \psi'$ and $\phi_i^{vio} \in \psi$, and iii) $\nexists \phi_i \in |\psi - \psi'|$ such that $\phi_i \in \phi^{hard}$.

iv) Construct self-loops such that $(s, g, a, s) \in E$ if $\exists$ (g, a) s.t. $g \subseteq g'$, $a \subseteq a'$ where $(s', g', a', s) \in E$ for some s' and $(s, g, a, s'') \notin E$ for any s''.

When the TAhd is completed, the initial plan is found by constructing the product automaton of the TAhd and the WTS following definition 10 and applying the modified Dijkstra Algorithm 1. The modifications consists of the inputs; initial time and current violation metrics. These inputs are used to set the distance metrics of the initial state and are all zero-valued for the initial offline planning. By allowing non-zero values the same algorithm can be used to re-plan when the mission has began and the time from start as well as violations are no longer zero when the graph search begins.

Algorithm 1: $dijkstraHD()$ Dijkstra Algorithm with Hybrid Distance as cost function

Data: P, h, τ_0, 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;
for $q \in Q$ **do**
  $dist(q) = d_h(q) = d_c(q) = d_d(q) = \infty$
  $pred(q) = \emptyset$;
$dist(q_0) = \tau_0$, $d_h(q_0) = d_h^0$, $d_c(q_0) = d_c^0$, $d_d(q_0) = d_d^0$;
while *no path found* **do**
  Pick $q \in SearchSet$ s.t. $q = \arg\min(d_h(q))$;
  if $q \in F$ **then** *path found*
  else
    for *every q' s.t. $q \rightsquigarrow q'$* **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 $dist(q')$, $d_h(q')$, $d_c(q')$, $d_d(q')$
        and $pred(q')$ and add q' to $SearchSet$;
  Remove q from $SearchSet$;
use $pred(q)$ to iteratively form the path back to q_0;
$\rightarrow r_{hd}^{min}$

In [11] we showed that a solution to the algorithm is always found under the assumption that the temporally bounded part of the MITL formula is feasible on the given WTS when deadlines are disregarded. The result holds here if we add the assumption that the hard constraint is feasible and does not contradict any eventually or until operators of the soft constraints when deadlines of the soft constraint are disregarded.

V. Learning Human Preference

In this section we consider Problem 2, i.e. learning the preferred value of h based on human control input u_h. The method is an inverse reinforcement learning (IRL) [16] approach and the estimated value of h is iteratively improved when new knowledge is given in the form of human input (i.e. when $u_h \neq 0$) under the assumption that the human is trying to help the system (i.e. u_h is chosen such that d_h is optimized for the true value of h). That is, $Cost(r_P^{t,*}, h^*) = \min_{r_P^t} Cost(r_P^t, h^*)$, where $r_P^{t,*}$ is the timed run of P which the human would guide the agent through (and the optimal run w.r.t. d_h given $h = h^*$), and $Cost(r_P^t, h) = d_h(proj(r_P^t, A_H), h)$ where $proj(r_P^t, A_H)$ is the projection of the timed run of the product automaton P onto the TAhd A_H as defined below. We also define the projection onto the WTS T for later use.

Definition 14: 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 $proj(r_P^t, T) = \pi_1, \pi_2, ..., \pi_m$.

To determine the k estimate of h we suggest solving

$$h_k = \arg\min_{h \in [0,1]} \sum_{i=1}^{k} p(Cost(r_P^{t,h}, h) - Cost(r_P^{t,i}, h)) \qquad (3)$$

where $p(x) = x$ if $x \leq 0$, and $p(x) = \infty$ if $x > 0$, $r_P^{t,h} = \arg\min_{r_P^t \in R_P^t} Cost(r_P^t, h)$, $r_P^{t,i}$, $i = 1, .., k$ are the previously suggested paths (i.e. $r_P^{t,1}$ is the initial plan and the outcome of Section IV), $R_P^t = \{r_P^t = q_1, q_2, ...q_m : r_P^{t,0} = q_1, q_2, ..., q_l, l \leq m\}$ and $r_P^{t,0}$ is the timed run of P which has been followed from start up until the time of the human input. That is, R_P^t is the set of timed runs of P which can be followed given the up-to-date trajectory. The function $p(x)$ is used to ensure that $Cost(r_P^{t,h_k}, h_k) \leq Cost(r_P^{t,i}, h_k)$ for $i = 1, ..., k$, removing any solutions h_k for which a previously suggested run would be better than the optimal run given the initial trajectory. No loss of correct solutions occurs if the assumption that u_h is chosen for a true h is correct. The solution to (3) is the h which maximizes how much d_h decreases due to the human input. The optimal timed run w.r.t. hybrid distance and h_k is then

$$r_P^{t,k+1} = \arg\min_{r_P^t \in R_P} Cost(r_P^t, h_k). \qquad (4)$$

The new path to follow is then found by the projection of $r_P^{t,k+1}$ onto the WTS, i.e, $r_{A_H}^{t,k+1} = proj(r_P^{t,k+1}, A_H)$. The solution to (3) and (4) can be found by implementing Algorithm 2.

VI. Re-planning for Collision Avoidance

In this section we consider collision avoidance. We will assume that the agents can share their current position with each other. Each agent determines if its next target region is occupied by another agent in which case it must stop and re-plan. The re-planning follows the steps; i) manipulate the weights of each state $q \in Q$ which correspond to the occupied region $\pi \in \Pi$ to make them deadlock states, ii) set the initial state q_0 to be the current state, iii) set the start time to the current time and iv) apply Algorithm 1 to the updated product. That is, we attempt to find an accepting path from the current state, which does not include the occupied state.

Algorithm 2: $irl4h()$: Finds h_k, $r_P^{t,k+1}$ and $r_{A_H}^{t,k+1}$

Data: $d_c(r_P^{t,i})$ and $d_d(r_P^{t,i})$ for $i = 0, ..., k$ and P
Result: h_k, $r_P^{t,k+1}$
for $h = 0, \delta, 2\delta, ..., 1$ **do**
  $dijkstraHD() \rightarrow r_P^{t,h}$ and h_d (Alg 1);
  $Cost(r_P^{t,h}, h) = h_d$;
  for $i = 1, ..., k$ **do**
    $Cost(r_P^{t,i}, h) = hd_c(r_P^{t,i}) + (1-h)d_d(r_P^{t,i})$;
$h_k = \arg\min \sum p(Cost(r_P^{t,0,h}, h) - Cost(r_P^{t,i}, h))$;
$dijkstraHD() \rightarrow r_P^{t,k+1}$ and h_d (Alg 1);

The steps above are iterated until either a new path is found or a maximum wait-time T_{wait}^{max} has passed. If the target state is still occupied after T_{wait}^{max} time units the agent attempts to find a solution to the temporary task 'move out of the way'. This is done by temporary updating the set of accepting states in the product automaton to include every neighbouring state which does not violate the hard constraint, and is not occupied. The temporary task can be solved if the hard constraint doesn't forbid all other transitions.

We denote the set of forbidden states (states which cannot reach the accepting states) as Q_T. Q_T can be determined indirectly by first finding $Q_T^{-1} = Q \backslash Q_T$ (the set of states which an accepting state can be reached from). Q_T^{-1} is found iteratively by: $q \in Q_T^{-1}$ if $q \rightsquigarrow q'$ and $q' \in Q_T^{-1}$, where initially $Q_T^{-1} = \mathcal{F}$. We can now apply the collision avoidance algorithm described in Algorithm 3, where we have made use of the second part of Definition 14.

Algorithm 3: $collAv()$ Collision Avoidance of agent i

Data: Position of agents: $x = x_1, x_2, ..., x_k$, current target state q_{next}, data for $dijsktraHD()$
while *no path found* **do**
  if $\exists j$ *s.t* $x_j \in \pi_{next}$ *where* $\pi_{next} = proj(q_{next}, T)$ **then**
    update τ_0, d_c^0, d_d^0, d_h^0 and Q^{init};
    set $d(q, q') = \infty\ \forall q$ if $proj(q', T) = \pi_{next}$;
    $dijkstraHD()$ (alg 1);
    if *no path is found* **then**
      wait for $t = \Delta T$;
      set $T_{wait} = T_{wait} + \Delta T$;
      check if π_{next} is free again;
      if $T_{wait} > T_{wait}^{max}$ **then**
        update τ_0 and set $q \in \mathcal{F}$ if $proj(q, T) = \pi$ where $\pi \in \Pi_{nb}$, and $q \notin Q_T$;
        $dijkstraHD()$ (alg 1);

VII. Restrictions on Human Control Input

We will now consider Problem 4, i.e, how to avoid violation of ϕ^{hard} when the human control input is non-zero. As in [15] we will use a mixed-initiative controller [14]:

$$u = u_r(x, \pi_s, \pi_g) + \kappa(x, \Pi)u_h(t) \qquad (5)$$

for each transition $(\pi_s, \pi_g) \in \rightarrow$, where u_r is the control input from the system designed to follow the plan which was conceived in Section IV-VI, and u_h is the human input. The problem then becomes to design κ such that ϕ^{hard} is never violated. To solve the problem we follow the same idea as in [15], namely to design κ such that: i) $\kappa = 0$ if $d_t = 0$, ii) $\kappa = 1$ if $d_t > d_s$, and iii) $\kappa \in [0, 1]$ and $\dot{\kappa} \propto \dot{d}_t$ if $0 < d_t \leq d_s$, where d_t is the minimum distance to a violating state and $d_s > 0$ is a safety margin. This was achieved in [15] by choosing:

$$\kappa(x, \mathcal{O}_t) = \frac{\rho(d_t - d_s)}{\rho(d_t - d_s) + \rho(\varepsilon + d_s - d_t)} \qquad (6)$$

where $\rho(s) = e^{-1/s}$ for > 0 and $\rho(s) = 0$ for $s \leq 0$, $\varepsilon > 0$ is a design parameter for safety, and $d_t = \min_{\pi \in \mathcal{O}_t} \|x - \pi\|$ where $\mathcal{O}_t$ contains all regions $\pi \in \Pi$ which corresponds to a violating state $q \in Q$. Unlike [15], here we must also consider the time constraints of ϕ^{hard}. Assuming that ϕ^{hard} has temporally bounded operators almost all states $\pi \in \Pi$ of the WTS will correspond to the violation of ϕ^{hard} for some time t (i.e. belong to $\mathcal{O}_t$). Hence, the solution in [15] is too conservative to apply here, setting $\kappa = 0$ in almost all states.

To solve this problem we use the set Q_T (containing all states which cannot reach an accepting state) which we constructed in the previous section, and construct a new set $Q_T^t = \{(q, t) : q \in Q_T, t = \min(x \in I_X^P(q))\}$ containing all states corresponding to the violation of ϕ^{hard} paired with the corresponding violated deadline (i.e. the minimum time required to enter the state). We then redefine: $d_t = \min_{(q,t) \in Q_T^t} dist(x, (q, t))$ where $dist(x, (q, t)) = \|x - proj(q, T)\|$ if $t_0 + d(\pi_0, proj(q, T)) > t$ and $dist(x, proj(q, T)) = \infty$ otherwise, where t_0 and π_0 are the time and state of the WTS at the time of calculation. The resulting d_t is then the minimum distance to a violating state, and hence equation (6) can be applied without the aforementioned issue.

VIII. Case Study

A simulation with two agents, each following the dynamics in eq. (7), has been performed. Agent 1 is partially controlled by a human user, i.e. u_1 follows eq. (5), while agent 2 is fully autonomous: $u_2 = u_r(x, \pi_s, \pi_g)$.

$$\dot{x}_i = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix} x_i + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u_i, \ i = 1, 2 \qquad (7)$$

Agent 1 is tasked with visiting the green areas, while agent 2 is tasked with visiting the blue areas, both with soft deadlines. Both agents should also try to avoid yellow areas while they are strictly forbidden to enter red areas. The resulting MITL specifications are $\phi_1 = \phi_1^{hard} \wedge \phi_1^{soft} = (\Box\neg a) \wedge (\Box\neg b \wedge \Diamond_{\leq 0.5} c \wedge \Diamond_{\leq 0.9} d)$ and $\phi_2 = \phi_2^{hard} \wedge \phi_2^{soft} = (\Box\neg a) \wedge (\Box\neg b \wedge \Diamond_{\leq 0.01} e \wedge \Diamond_{\leq 0.03} f)$. The control input u_r and the transition times (which are over-approximations) were determined following [18]. Since the violation distances used during planning depends on the transition times it follows that they are over-approximations. During the online

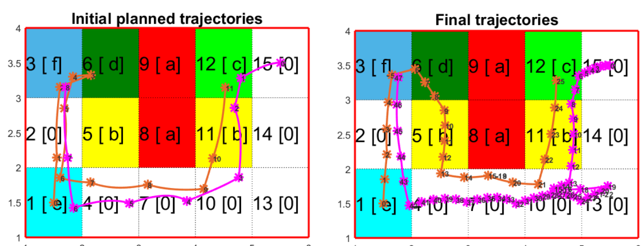
TABLE II: Values of the violation distances: d_c, d_d and d_h for agent 1 and agent 2 in the case study.

		Initial Plan/Trajectory		Final Plan/Trajectory	
		Estimates	Real Values	Estimates	Real Values
Ag. 1	d_c	0.0418	0.0405	0	0
	d_d	0.0418	0.0405	0.1137	0.1037
	d_h	0.0418	0.0405	0	0
Ag. 2	d_c	0.9351	0.7573	1.6571	1.2795
	d_d	0.0719	0.0664	0.0759	0.0668
	d_h	0.7099	0.4118	0.8665	0.6731

re-planning the real-time violation distances are used for tracking d_h while the approximations are used for planning.

The workspace and the initial plans for each agent is illustrated in Figure 1a. During the online run the human user has a chance to apply control input every 0.015 time steps. Figure 1b illustrates one possible outcome. Notable events are: 1) The human steers agent 1 into the yellow region 5 (instead of region 3) during step 6-9, indicating that h should be increased. Agent 1 determines that $h = 1$ is the optimal choice. 2) The agents block each other at step 15. After failing to re-plan and wait agent 2 apply a temporary task and move into state 13 allowing agent 1 to pass.

The values of the violation distances are given in Table II. As expected the real distances are smaller than the estimates. Comparing the initial plans with the final plans, d_h decreases for agent 1 due to the correct value of h being discovered, while it increases for agent 2 since the collision avoidance forces the agent to take a longer route.



(a) Initial plans of the 2 agents in the case study.

(b) Final trajectories of agent 1 and 2 in the case study.

Fig. 1: Agent 1 follows the orange trajectory and agent 2 follows the magenta trajectory. Each number/star along the trajectories indicates one iteration where the human had a chance to change her control input, the time step in between is 0.015 time units.

The simulation was performed in Matlab and was executed as a turn-taking game. The individual graph search processes were performed in 0.01-0.06 s on a laptop with a Core i7-6600U 2.80 GHz processor. The h-learning algorithm (Alg. 2) requires the graph search algorithm to run multiple times (inversely proportional to the choice of step size δ).

IX. CONCLUSIONS AND FUTURE WORK

We have presented a decentralized control synthesis framework for a multi-agent system under hard and soft constraints given as MITL specifications. The framework uses mixed initiative control to allow a human to affect the trajectories of the agents while guaranteeing satisfaction for the hard constraints. The human input is used in an IRL approach to learn the human preference considering the manner of violation of the soft constraints. A collision avoidance algorithm is used to ensure safety. The result is a control policy which guarantees satisfaction of hard constraints and maximizes the satisfaction of soft constraints with respect to human preference, while avoiding collisions.

Future work includes determining under which conditions agents should re-plan to optimize performance time, determining how the step size of h in the learning algorithm can be optimized, performing simulations with a larger number of agents, and implementing the framework on a robotic platform in real-time.

REFERENCES

[1] C. Belta, B. Yordanov, and E. A. Gol, *Formal methods for discrete-time dynamical systems*. Springer, 2017, vol. 89.

[2] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, Dec 2009.

[3] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343 – 352, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000510980800455X

[4] J. Ouaknine and J. Worrell, "On the decidability of metric temporal logic," in *Logic in Computer Science, 2005. LICS 2005. Proceedings. 20th Annual IEEE Symposium on*. IEEE, 2005, pp. 188–197.

[5] D. Souza and P. Prabhakar, "On the expressiveness of mtl in the pointwise and continuous semantics," *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 1, pp. 1–4, 2007.

[6] D. Ničković and N. Piterman, "From mtl to deterministic timed automata," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2010, pp. 152–167.

[7] E. A. Gol and C. Belta, "Time-constrained temporal logic control of multi-affine systems," *Nonlinear Analysis: Hybrid Systems*, vol. 10, pp. 21–33, 2013.

[8] P.-J. Meyer and D. V. Dimarogonas, "Compositional abstraction refinement for control synthesis," *Nonlinear Analysis: Hybrid Systems*, 2017, to appear.

[9] G. E. Fainekos, "Revising temporal logic specifications for motion planning," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 40–45.

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

[11] S. Andersson and D. V. Dimarogonas, "Human in the Loop Least Violating Robot Control Synthesis under Metric Interval Temporal Logic Specifications," *European Control Conference (ECC) 2018*, 2018.

[12] S. Carr, N. Jansen, R. Wimmer, J. Fu, and U. Topcu, "Human-in-the-loop synthesis for partially observable markov decision processes," in *2018 Annual American Control Conference (ACC)*, June 2018, pp. 762–769.

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

[14] W. Li, D. Sadigh, S. S. Sastry, and S. A. Seshia, "Synthesis for human-in-the-loop control systems," in *Tools and Algorithms for the Construction and Analysis of Systems*, E. Ábrahám and K. Havelund, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 470–484.

[15] M. Guo, S. Andersson, and D. V. Dimarogonas, "Human-in-the-loop mixed-initiative control under temporal tasks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6395–6400.

[16] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, 2000, pp. 663–670.

[17] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

[18] S. Andersson, A. Nikou, and D. V. Dimarogonas, "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)*, 2017.