Minimal Specification Revision for Weighted Transition Systems

Kangjin Kim and Georgios Fainekos

***Abstract*— In this paper, we study the problem of revising Linear Temporal Logic (LTL) formulas that capture specifications for optimal planning over weighted transition systems. Namely, it is assumed that the model of the system is a weighted finite state transition system. The LTL specification captures the system requirements which must be satisfied by a plan which costs less than a certain cost budget. If the cost bounds cannot be satisfied with the initial specification, then it is desirable to return to the user a specification that can be satisfied on the system within the desired cost budget. We prove that the specification revision problem for automata-based optimal planning is NP-complete. In order to provide exact solutions to the problem, we present an Integer Linear Program (ILP) and a Mixed-Integer Linear Program (MILP) formulation for different versions of the problem. Finally, we indicate that a Linear Program (LP) relaxation can compute fast approximations to the problem.**

I. Introduction

Temporal logics have become a popular formalism for capturing high level specifications for both single robots and teams of robots. The advantage of temporal logics is twofold. First, they provide a good interface with natural languages [1], [2]. Second, they provide a mathematical formal framework for combining high-level reasoning with low-level controllers with provable results and guaranteed performance of the final control system. In particular, Linear Temporal Logic (LTL) has been utilized as a specification language in a range of robotics applications (see [3]–[13]).

In order for temporal logic planning to become a viable framework for robot control several aspects of human-robot interaction must be resolved. Besides the human instructing the robot, the robot must be able to provide feedback to the user if it cannot execute some user commands. Along these lines, in our previous work [14], we introduced the specification revision problem for Linear Temporal Logic (LTL). When the LTL planning phase fails, the specification revision problem attempts to determine a new specification which is close to the initial user intent and, moreover, it is satisfiable by the system within its environment. In the follow-up paper [15], we studied the problem when the specifications are provided as ω-automata and we proved that the problem is computationally hard. In view of this negative result, we developed a polynomial-time heuristic approximation algorithm with a guaranteed approximation ratio in some special cases [16].

In this paper, we investigate the problem of specification revision for optimal planning with LTL specifications. For instance, in optimal LTL planning [11], [17], each action of the system incurs some cost and the goal is to compute a plan that satisfies the LTL specification with minimal cost. Here, we pose the question what happens if we impose some cost constraints for our plan and the minimal cost plan that we compute does not satisfy these constraints. Since the environment and system cannot be modified, we must compute an alternative specification that remains as close as possible to the initial user intent. In other words, we attempt to compute the largest sub-specification that can be satisfied on the system and the resulting plan is satisfiable with total cost less than our cost constraints. We prove that the problem is computationally hard and we provide a formulation of the problem as a (mixed-) integer linear program (M)ILP. Also, we provide heuristics in order to provide a linear programming relaxation that can provide fast approximations to the optimal solution.

Related research: A related problem is the detection of the causes of unrealizability in LTL games. In this case, a number of heuristics have been developed in order to localize the error and provide meaningful information to the user for debugging [18], [19]. Along these lines, LTLMop [20] was developed to debug unrealizable LTL specifications in reactive planning for robotic applications. In the context of general planners, the problem of finding good excuses on why the planning failed has been studied in [21]. Over-Subscription Planning (OSP) [22] and Partial Satisfaction Planning (PSP) [23] are also very related problems. OSP determines a proper subset of a well-defined, but oversubscribed, conjunctive goal to meet time and energy limitations. PSP is the problem where each goal is a soft constraint and the planner attempts to find a good quality plan for a subset of the goals. In terms of MILP formulation, the specification revision problem bares some similarities with the vehicle routing problem with refueling constraints [24].

II. Problem Formulation

In this paper, we work with discrete abstractions (Finite State Machines) of the continuous robotic control system [3]. This is a common practice in approaches that hierarchically decompose the control synthesis problem into high-level discrete planning synthesis and low level continuous feedback controller composition [3], [6]. Each state of the Finite State Machine (FSM) $\mathcal{T}$ is labeled by a number of symbols from a set $\Pi = \{\pi_0, \pi_1, \ldots, \pi_n\}$ that represent regions in the workspace or the configuration space of the robot, or more generally, actions that can be performed by the robot. Moreover, each state of the FSM has an associated weight. The weights can represent the worst case energy or

This work has been partially supported by award NSF CNS 1116136.

K. Kim and G. Fainekos are with the School of Computing, Informatics and Decision Systems Engineering, Arizona State University, Tempe, AZ 85281, USA {Kangjin.Kim,fainekos}@asu.edu

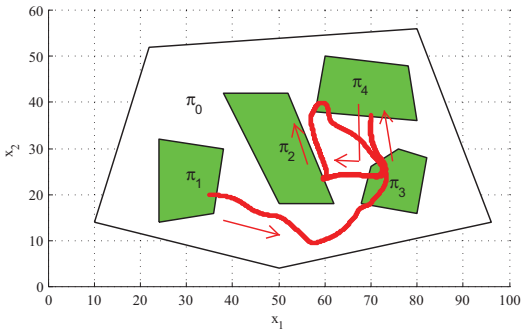

Fig. 1. Example 1: The environment and a trajectory of system (1) that satisfies the specification ϕ_{e1}.

time needed for traversing a region. In general, the weights on the FSM capture some quantitative values on the cost of performing some action.

The control requirements for such a system can be posed in propositional Linear Temporal Logic (LTL) [25] or, more generally, using specification automata $\mathcal{B}$ with Büchi acceptance conditions [26] also known as ω-automata. LTL formulas are built over a set of atoms, the set Π in our case, using combinations of the traditional and temporal operators. Traditional logic operators are the *conjunction* ($\wedge$), *disjunction* ($\vee$), *negation* ($\neg$). Some of the temporal operators are *eventually* ($\Diamond$), *always* ($\Box$) and *until* ($\mathcal{U}$). LTL can describe the usual properties of interest for control problems, i.e. *reachability* ($\Diamond\pi$) and *safety:* ($\Box\pi$ or $\Box\neg\pi$). Beyond the usual properties, LTL can capture sequences of events and certain infinite behaviors. For example, *sequencing:* The requirement that we must visit π_1, π_2 and π_3 in that order is naturally captured by the formula $\Diamond(\pi_1 \wedge \Diamond(\pi_2 \wedge \Diamond\pi_3))$. More complicated specifications can be composed from the basic specifications using the logic operators (e.g., [3], [6]).

We assume that we use only formulas in Negation Normal Form (NNF) and that each negated atomic proposition $\neg\pi$ has been replaced by a new symbol, e.g. $\overline{\pi}$, which is added to Π. The details on why LTL in NNF is equivalent to full LTL can be found in [3]. Any LTL formula can be converted into an ω-automaton [27].

The following example, which is the running example of the paper, presents such a typical scenario for motion planning of a mobile robot.

Example 1 (Robot Motion Planning): We consider a mobile robot which operates in a planar environment. The continuous state variable $x(t)$ models the internal dynamics of the robot whereas only its position $y(t)$ is observed. In this example, we will consider a 2nd order model of the motion of a planar robot (dynamic model):

$$\dot{x}_1(t) = x_2(t), \quad \dot{x}_2(t) = u(t), \quad y(t) = x_1(t) \qquad (1)$$

where $x_1(t) \in \mathbb{R}^2$, $x_1(0) \in X_{1,0}$, $x_2(t) \in \mathbb{R}^2$, $x_2(0) = 0$, $u(t) \in U$. The robot is moving in a convex polygonal environment with four areas of interest denoted by $\pi_1, \pi_2, \pi_3, \pi_4$ (see Fig. 1). Initially, the robot is placed somewhere in the region labeled by π_1. The robot must accomplish the task: "Periodically visit areas π_2, π_3 and π_4 with the additional requirement that at least one π_4 should be followed by at least one π_2". The informal constraint is captured by the LTL specification $\phi_{e1} = \Box(\Diamond\pi_3 \wedge \Diamond(\pi_4 \wedge \Diamond\pi_2))$.

The LTL planning is performed on a discrete abstraction of the environment (FSM) which results from a cell decomposition. The weights that label the transitions of the FSM are proportional to the area of the cell at the source of the edge. The weights capture the worst case energy or time needed to traverse the cell. Figure 1 presents a trajectory that satisfies the specification ϕ_{e1}. The trajectory was created using the hierarchical control framework from [3]. The trajectory consists of a transient behavior (prefix) that starts in π_1 and visits first π_3 and then π_4 and π_2 and a limit cycle (loop) that continuously visits π_4, π_3 and π_2. $\triangle$

A challenging problem arises when there is a hard constraint on the total cost for either the transient part or the stable part of the discrete trajectory. Such constraints could be imposed if for example there is a total budget on the energy storage and one of the atomic propositions represents a recharging station. In other words, even though the specification is satisfiable on the system if the cost constraint is ignored, it becomes unsatisfiable when considering the hard cost constraints. If the environment or the robot capabilities cannot be modified, then the goal is to detect the largest sub-specification that can be actually realized by the robot.

Problem 1 (Weighted Minimal Revision Prob. (WMRP)): Given a weighted FTS $\mathcal{T}$, an LTL formula ϕ and a cost budget $C \in \mathbb{R}_{\geq 0}$, if the specification ϕ cannot be satisfied on $\mathcal{T}$ under cost C, then find the "closest" specification ϕ' to ϕ which is satisfied on $\mathcal{T}$ with cost less than C.

The unweighted version of Problem 1 was introduced and studied in [14], [15]. As indicated in [14], the specification revision problem is easy when the discrete controller synthesis phase fails due to unreachable states in the system. Thus, in this paper, we concentrate on the harder problem of minimal revision when all the states on $\mathcal{T}$ are reachable.

Assumption 1: All the states on $\mathcal{T}$ are reachable.

Contributions: In this paper, we formally define WMRP and we show that the problem is NP-complete if we assume that input is provided as a specification automaton. Also, we provide an Integer Linear Program (ILP) and a Mixed-Integer Linear Program (MILP) formulation for two different versions of WMRP. An LP relaxation is considered and some examples are presented.

III. Optimal Discrete Controllers

In this section, we provide a brief review of automata based optimal planning over weighted transition systems. First, we will present how the cost of a periodic discrete plan can be optimized and, then, we will briefly comment on the more challenging optimal control problem with LTL constraints presented in [11], [17].

In order to use temporal logics to specify requirements for continuous systems, we need to construct a finite partition of the robot's workspace (or configuration space). We can

use many efficient cell decomposition methods for polygonal environments [28]. This results in a topological graph $G = (Q, E)$ which describes which cells are topologically adjacent, i.e., each node $q \in Q$ in the graph represents a cell and each edge $e = (q, q') \in E$ in the graph implies topological adjacency of the cells. Each such cell will be a state in the FSM which will be labeled by a weight and by one or more atomic propositions from Π. Next, we formally define the FSM that can be constructed from the graph G.

Definition 1 (FSM): A Finite State Machine is a tuple $\mathcal{T} = (Q, Q_0, \rightarrow_{\mathcal{T}}, h_{\mathcal{T}}, w, \Pi)$ where: Q is a set of states; $Q_0 \subseteq Q$ is the set of possible initial states; $\rightarrow_{\mathcal{T}} = E \subseteq Q \times Q$ is the transition relation; $h_{\mathcal{T}} : Q \rightarrow \mathcal{P}(\Pi)$ maps each state q to the set of atomic propositions that are true on q; and, $w : Q \rightarrow \mathbb{R}_{\geq 0}$ returns the weight in each state.

We define a *path* on the FSM to be a sequence of states and a *trace* to be the corresponding sequence of sets of propositions. Formally, a path is a function $p : \mathbb{N} \rightarrow Q$ such that for each $i \in \mathbb{N}$ we have $p(i) \rightarrow_{\mathcal{T}} p(i+1)$ and the corresponding trace is the function composition $\bar{p} = h_{\mathcal{T}} \circ p : \mathbb{N} \rightarrow \mathcal{P}(\Pi)$. The language $\mathcal{L}(\mathcal{T})$ of $\mathcal{T}$ consists of all possible traces.

In this work, we are interested in the construction of automata that only accept the traces of $\mathcal{T}$ which satisfy the LTL formula ϕ. Such automata (which we will refer to as specification or Büchi automata [29]) differ from the classic finite automata in that they accept infinite strings (traces of $\mathcal{T}$ in our case).

Definition 2 (Automaton): An automaton is a tuple $\mathcal{B} = (S_{\mathcal{B}}, s_0^{\mathcal{B}}, \Omega, \delta_{\mathcal{B}}, F_{\mathcal{B}})$ where: $S_{\mathcal{B}}$ is a finite set of states; $s_0^{\mathcal{B}}$ is the initial state; Ω is an input alphabet; $\delta_{\mathcal{B}} : S_{\mathcal{B}} \times \Omega \rightarrow \mathcal{P}(S_{\mathcal{B}})$ is a transition function; and $F_{\mathcal{B}} \subseteq S_{\mathcal{B}}$ is a set of final states.

When $s' \in \delta_{\mathcal{B}}(s, l)$, we also write $s \overset{l}{\rightarrow}_{\mathcal{B}} s'$ or $(s, l, s') \in \rightarrow_{\mathcal{B}}$. A *run* r of $\mathcal{B}$ is a sequence of states $r : \mathbb{N} \rightarrow S_{\mathcal{B}}$ that occurs under an input trace $\bar{p}$ taking values in Ω. That is, for $i = 0$ we have $r(0) = s_0^{\mathcal{B}}$ and for all $i \geq 0$ we have $r(i) \overset{\bar{p}(i)}{\rightarrow}_{\mathcal{B}} r(i+1)$. Let $\lim(\cdot)$ be the function that returns the set of states that are encountered infinitely often in the run r of $\mathcal{B}$. Then, a run r of a Büchi automaton $\mathcal{B}$ over an infinite trace $\bar{p}$ is *accepting* if and only if $\lim(r) \cap F_{\mathcal{B}} \neq \emptyset$. Finally, we define the language $\mathcal{L}(\mathcal{B})$ of $\mathcal{B}$ to be the set of all traces $\bar{p}$ that have a run that is accepted by $\mathcal{B}$.

A *specification* automaton is a Büchi automaton where the input alphabet is the powerset of the labels of the system $\mathcal{T}$, i.e., $\Omega = \mathcal{P}(\Pi)$. In order to simplify the discussion in Section IV, we will be using the following assumptions and notation

- we define the set $E_{\mathcal{B}} \subseteq S_{\mathcal{B}} \times S_{\mathcal{B}}$, such that $(s, s') \in E_{\mathcal{B}}$ iff $\exists l \in \Omega \; . \; s \overset{l}{\rightarrow}_{\mathcal{B}} s'$; and,
- we define the function $\lambda_{\mathcal{B}} : S_{\mathcal{B}} \times S_{\mathcal{B}} \rightarrow \Omega$ which maps a pair of states to the label of the corresponding transition, i.e., if $s \overset{l}{\rightarrow}_{\mathcal{B}} s'$, then $\lambda_{\mathcal{B}}(s, s') = l$; and if $(s, s') \notin E_{\mathcal{B}}$, then $\lambda_{\mathcal{B}}(s, s') = \emptyset$.

In brief, our goal is to generate paths on $\mathcal{T}$ that satisfy the specification $\mathcal{B}_\phi$. In automata theoretic terms, we want to find the subset of the language $\mathcal{L}(\mathcal{T})$ which also belongs to the language $\mathcal{L}(\mathcal{B}_\phi)$. This subset is simply the intersection of the two languages $\mathcal{L}(\mathcal{T}) \cap \mathcal{L}(\mathcal{B}_\phi)$ and it can be constructed by taking the product $\mathcal{T} \times \mathcal{B}_\phi$ of the FSM $\mathcal{T}$ and the specification automaton $\mathcal{B}_\phi$. Then, given an initial state in the FSM $\mathcal{T}$, we can choose a particular trace from $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{T} \times \mathcal{B}_\phi) = \mathcal{L}(\mathcal{T}) \cap \mathcal{L}(\mathcal{B}_\phi)$ according to a preferred criterion.

Definition 3: The product automaton $\mathcal{A} = \mathcal{T} \times \mathcal{B}_\phi$ is the automaton $\mathcal{A} = (S_{\mathcal{A}}, s_0^{\mathcal{A}}, \mathcal{P}(\Pi), \delta_{\mathcal{A}}, F_{\mathcal{A}})$ where:

- $\mathcal{S}_{\mathcal{A}} = Q \times S_{\mathcal{B}_\phi}$,
- $s_0^{\mathcal{A}} = \{(q_0, s_0^{\mathcal{B}_\phi}) \mid q_0 \in Q_0\}$,
- $\delta_{\mathcal{A}} : S_{\mathcal{A}} \times \mathcal{P}(\Pi) \rightarrow \mathcal{P}(S_{\mathcal{A}})$ s.t. $(q_j, s_j) \in \delta_{\mathcal{A}}((q_i, s_i), l)$ iff $q_i \rightarrow_{\mathcal{T}} q_j$ and $s_j \in \delta_{\mathcal{B}_\phi}(s_i, l)$ with $l \subseteq h_{\mathcal{T}}(q_j)$,
- $F_{\mathcal{A}} = Q \times F_{\mathcal{B}}$ is the set of accepting states.

We say that $\mathcal{B}_\phi$ is *satisfiable* on $\mathcal{T}$ if $\mathcal{L}(\mathcal{A}) \neq \emptyset$.

The class of languages accepted by Büchi automata can be fully characterized by ultimately periodic traces [30]. That is traces of the form: there exist constants $m, n \in \mathbb{N}$ s.t. for all $k \geq 0$, we have $\bar{p}(m+k) = \bar{p}(m + \mod(k, n))$, where $\mod$ is the modulo operation. In other words, an ultimately periodic trace consists of a *finite* (or *transient* or *prefix*) part $\bar{p}(0)\bar{p}(1)\ldots\bar{p}(m-1)$ and a loop (or *lasso* or *steady state* or *suffix*) part $\bar{p}(m)\bar{p}(m+1)\ldots\bar{p}(m+n-1)$ repeated ad infinitum. Based on that observation, we can heuristically construct a cost function that will help us compute a plan that satisfies our overall cost bound C:

$$\mathcal{C}_1(p) = \max\left\{\sum_{i=1}^{m}(w(p(1+i))), \sum_{i=1}^{n}(w(p(m+i)))\right\}.$$

The first quantity is the cost of the path to get to the state $p(m)$, while the second is the cost of the steady state.

Based on the previous cost function, it is easy to compute the optimal path on $\mathcal{T}$ that satisfies ϕ. We define a weight function $w_{\mathcal{A}}$ on the edges of $\mathcal{A}$ as follows: $\forall \; (q_j, s_j) \in \delta_{\mathcal{A}}((q_i, s_i), l), \; w_{\mathcal{A}}((q_i, s_i), (q_j, s_j)) = w(q_i)$. Then we run Dijkstra's shortest path algorithm [31] to compute the shortest path from the initial state to any accepting state in $F_{\mathcal{A}}$. Then, starting from each reachable accepting state, we compute the shortest path to get back to that state. Finally, we can select a path that minimizes $\mathcal{C}_1(p)$.

Example 2: In Example 1, $\mathcal{T}$ has 44 states and the Büchi automaton has 6 states. The product automaton $\mathcal{A}$ has 270 states. The minimal cost planning method proposed above took 0.63sec on an i7 at 2.93GHz with 8GB on a prototype Matlab implementation. Both the transient and the stable trajectories are generated such that total weight of the paths on the discrete abstraction is minimal. The costs of the prefix and loop are 855 and 1015, respectively. $\triangle$

The procedure described above is meaningful only in certain motion planning scenarios. For example, such a planning framework can be useful in cases where the vehicle can continuously recharge, e.g., using solar panels, or regenerate energy while operating. In these cases, it is desirable that the cost of the periodic part of the plan as well as the transient behavior have cost less than cost budget C which could indicate depleted power sources.

A different LTL optimal planning framework has been developed in [11], [17]. In [17], the authors solve the optimal

planning problem for specifications of the form

$$\phi := \varphi \wedge \square\diamond\psi, \tag{2}$$

where ψ is a Boolean combination of atomic propositions that must be satisfied infinitely often. For instance, we could set $\psi = \pi^\star$ where $\pi^\star$ is an atomic proposition indicating a recharging or time reseting operation in a particular location in the environment. Then, the optimal control framework attempts to compute paths that when passing through spi have cost less than C.

In the following, we will assume that ψ is a single atomic proposition, i.e., $\psi = \pi^\star$. The discussion can be generalized to Boolean combinations of atomic propositions. Formally, given a path p, the function $\alpha_p^{\pi^\star} : \mathbb{N} \to \mathbb{N}$ returns the i-th appearance of $\pi^\star$ in $\bar{p}$. The cost $\mathcal{C}(p)$ of a path is defined as

$$\mathcal{C}_2(p) = \limsup_{i \to +\infty} \sum_{j=\alpha(i)}^{\alpha(i+1)} w(p(j))$$

$\mathcal{C}_2(p)$ is finite only if $\pi^\star$ occurs periodically in the trajectory. The main results from [17] can be summarized as follows.

Theorem 1: There exists $\bar{p} \in \mathcal{L}(\mathcal{A})$ s.t. $\bar{p}$ is ultimately periodic and the corresponding path p minimizes the cost function $\mathcal{C}_2(p)$.

The computation of a path that minimizes $\mathcal{C}_2$ is similar to the algorithm that produces plans that minimize $\mathcal{C}_1$, but with a major difference. There is a set of nodes R – potentially different from $F_\mathcal{A}$ – that are labeled by $\pi^\star$ and when visited they reset the cost of a path. A detailed description of the algorithm appears in [17].

The paths computed with the above process are guaranteed to be the minimum cost paths. In realistic scenarios, it is to be expected to have hard constraints on the allowable worst case costs. As an example consider a scenario where the costs represent worst case energy consumption in order to traverse a region and the special atomic proposition indicates the location of a charging unit. In such scenarios, it is necessary to revise the mission plan requirements in order to identify a feasible plan which satisfies the hard cost constraints.

IV. The LTL Revision Problem

The specification revision problem concerns the search for one or more specifications which are related to the initial user requirement and, which, furthermore, can be satisfied on the weighted transition system under some hard cost constraints. One of the fundamental questions regards the form of the search space of the specifications. Since the initial user specification allows only system behaviors with higher cost than the constraint, it is natural to relax some of the requirements in order to permit more behaviors and, hopefully, find a behavior with lower cost.

The unconstrained LTL formula search space for a revised specification is

$$\mathfrak{F}_\mathcal{T}^C = \{\phi \in LTL(\Pi) \mid \exists \bar{p}.\bar{p} \in \mathcal{L}(\mathcal{T} \times \mathcal{B}_\phi) \wedge \mathcal{C}(p) \leq C\},$$

where $\mathcal{C}$ is $\mathcal{C}_1$ or $\mathcal{C}_2$. However, as we have demonstrated through examples in [14] for the unweighted version of the problem, $\mathfrak{F}_\mathcal{T}^C$ also contains specifications that from the user perspective cannot be considered valid specification revisions. Thus, we must impose some constraints on the search space. First, we define an ordering relation over the set of LTL formulas.

Definition 4: Let $\phi, \psi \in LTL(\Pi)$, then we define $\phi \preceq \psi$ if and only if $\mathcal{L}(\phi) \subseteq \mathcal{L}(\psi)$.

We define the set of *ultra relaxations* as follows.

Definition 5 (Ultra Relaxation): Let $\phi \in LTL(\Pi)$, the set $\mathfrak{UR}(\phi)$ of all valid formula relaxations of ϕ can be constructed using the recursive operator $\mathbf{rel}(\phi)$ as follows:

$$\mathbf{rel}(\pi) \in \{\pi, \top\}, \qquad \text{OP}_1\phi = \text{OP}_1\mathbf{rel}(\phi)$$
$$\mathbf{rel}(\phi_1 \ \text{OP}_2\phi_2) = \mathbf{rel}(\phi_1) \ \text{OP}_2\mathbf{rel}(\phi_2)$$

where OP_1 is any unary and OP_2 is any binary operator.

Informally, a valid formula relaxation is one that recursively relaxes each atomic proposition π of the initial specification ϕ.

Example 3: Let us consider the specification ϕ_{e1} of Example 1. Then, $|\mathfrak{UR}(\phi_{e1})| = 2^3 = 8$. As an example, we have $\varphi = \square(\diamond\pi_3 \wedge \diamond(\pi_4 \wedge \diamond\top)) \in \mathfrak{UR}(\phi_{e1})$. Note though that φ is equivalent to $\square(\diamond\pi_3 \wedge \diamond\pi_4)$. For clarity in the presentation, we will be using equivalent formulas interchangeably to refer to formulas in $\mathfrak{UR}(\cdot)$.

The following result is immediate from Theorem 1 in [14].

Corollary 1: For any $\phi \in LTL(\Pi)$ and $\phi' \in \mathfrak{UR}(\phi)$, we have $\phi \preceq \phi'$.

Therefore, a restricted version of Problem 1 can be formally restated as:

Problem 2: Given a system $\mathcal{T}$, a specification ϕ and a cost $C \in \mathbb{R}_{\geq 0}$ such that $\phi \notin \mathfrak{F}_\mathcal{T}^C$, find $\varphi \in \mathfrak{F}_\mathcal{T}^C \cap \mathfrak{UR}(\phi)$ such that for any other $\psi \in \mathfrak{F}_\mathcal{T}^C \cap \mathfrak{UR}(\phi)$, we have $\psi \not\preceq \varphi$.

Remark 1: A restricted search space and, thus, a restricted version of Problem 1 are necessary for two reasons. First, the revised specification must be related to the initial user intent rather than include arbitrary requirements (see [14]). Second, as we will demonstrate in the next section, the restricted version of the problem is already computationally hard.

Obviously, with the aforementioned restrictions the WMRP is decidable. For a formula ϕ, there is a finite number of revisions in $\mathfrak{UR}(\phi)$ that we must consider. For each $\phi' \in \mathfrak{UR}(\phi)$, we can solve the optimal path planning problem and, then, choose the revised specification with the least modifications that produces optimal paths with cost less than the bound C. Nevertheless, typical examples of LTL specifications can have 10-30 occurrences of atomic propositions in a formula (see [17] for an interesting collection). This means that the optimal LTL planning problem (which includes the Büchi automaton synthesis for each new formula) must be solved anywhere from 1,000 times to 1 billion times. Next, we study the question whether the problem really requires exploring all the combinations for the optimal solution.

V. LTL Revision as a Shortest Path Problem

In this section, we present how Problem 2 can be posed as a shortest path problem on a weighted labeled graph.

Without loss of generality, we will assume that for any given specification ϕ, each atomic proposition π in ϕ appears only once in ϕ. If this is not the case, then for each additional occurrence of π in ϕ, we can replace it with a new atomic proposition, add the proposition to Π and modify the map $h_{\mathcal{T}}$ accordingly. This change is necessary in order to uniquely identify in ϕ which propositions need to be replaced by $\top$.

Given a specification ϕ, we can construct the corresponding specification automaton $\mathcal{B}_\phi$. Using the weighted labeled transition system $\mathcal{T}$ and the specification automaton $\mathcal{B}_\phi$, we can construct a graph $G_{\mathcal{A}}$ which corresponds to the product automaton $\mathcal{A}$ while considering the effect of revisions and the weights.

Definition 6: Given a system $\mathcal{T}$ and a specification automaton $\mathcal{B}_\phi$, we define the graph $G_{\mathcal{A}} = (V, E, v_s, V_f, W, L, R)$, which corresponds to the product $\mathcal{A} = \mathcal{T} \times \mathcal{B}_\phi$ as

- $V = \mathcal{S}_{\mathcal{A}}$ is the set of nodes
- $E = E_{\mathcal{A}} \cup E_D \subseteq \mathcal{S}_{\mathcal{A}} \times \mathcal{S}_{\mathcal{A}}$, where
  - $E_{\mathcal{A}}$ is the set of edges that correspond to transitions on $\mathcal{A}$, i.e., $((q,s),(q',s')) \in E_{\mathcal{A}}$ iff $\exists l \in \mathcal{P}(\Pi)$. $(q,s) \xrightarrow{l}_{\mathcal{A}} (q',s')$; and
  - E_D is the set of edges that correspond to disabled transitions, i.e., $((q,s),(q',s')) \in E_D$ iff $q \rightarrow_{\mathcal{T}} q'$ and $s \xrightarrow{l}_{\mathcal{B}_\phi} s'$ with $l \cap (\Pi - h_{\mathcal{T}}(q')) \neq \emptyset$.
- $v_s = s_0^{\mathcal{A}}$ is the source node,
- $V_f = F_{\mathcal{A}}$ is the set of sinks,
- $W : E \rightarrow \mathbb{R}_{\geq 0}$ assigns a weight to each edge in E. If $e = ((q,s),(q',s')) \in E$, then $w(e) = w(q)$.
- $L : E \rightarrow \mathcal{P}(\Pi)$ maps each edge of the graph to the set of symbols that need to be removed in order to enable the edge in the product automaton $\mathcal{A}$. If $e = ((q,s),(q',s')) \in E$, then $L(e) = \lambda_{\mathcal{B}_\phi}(s,s') - h_{\mathcal{T}}(q')$.
- $R = \{(q,s) \in V \mid \pi^\star \in h_{\mathcal{T}}(q)\}$ is the set of "cost reset" nodes.

We remark that the graph $G_{\mathcal{A}}$ is essentially the same graph as the graph of $\mathcal{A}$ with the addition of the disabled edges due to the specification constraints. Therefore, any path on the graph of $\mathcal{A}$ appears as a path on $G_{\mathcal{A}}$.

A path $\eta = v_0 v_1 v_2 \dots v_n$ on $G_{\mathcal{A}}$ is a sequence of nodes that start from the source $v_0 = v_s$, follow the edges from E, i.e., for $0 \leq i < n$, $(v_i, v_{i+1}) \in E$, and end in one of the sinks $v_n \in V_f$. The cost of the corresponding path is defined as $C_{G_{\mathcal{A}}}(\eta) = \langle |L(\eta)|, W(\eta) \rangle$. $L(\eta) = \bigcup_{i=0}^{|\eta|-1} L(v_i, v_{i+1})$ is the set of symbols that need to be removed for the path to become enabled on $\mathcal{A}$. $W(\eta) = \max_{j=0,1,\dots,k-1} \sum_{i=r_j}^{r_{j+1}} W(v_i, v_{i+1})$ is the weight of the path after $k-1$ cost resets by visiting a node in R. Here, $r_0 r_1 \dots r_k$ with $r_i \in \{0, 1, \dots, |\eta| - 1\}$ is the sequence of indices such that $v_{r_i} \in R$, $r_0 = 0$ and $r_k = |\eta| - 1$. In the special case where there are no cost resets, i.e., $k = 1$, then $W(\eta) = \sum_{i=0}^{|\eta|-1} W(v_i, v_{i+1})$. Then, by construction, Problem 2 is reduced to solving a number of the following problems.

Problem 3: Given a weighted labeled graph $G_{\mathcal{A}}$ as in Def. 6 and a cost bound $K \in \mathbb{R}_{\geq 0}$, find a path η on the graph such that $|L(\eta)|$ is minimum over all paths in $G_{\mathcal{A}}$ while $W(\eta) \leq K$.

We refer to the last problem as Constrained Minimally Labeled Path (CMLP). It is easy to show that the corresponding decision problem is NP-complete.

Theorem 2: Given an instance of the CMLP $(G_{\mathcal{A}}, K)$ and a bound Λ, the decision problem of whether there exists a path η such that $|L(\eta)| \leq \Lambda$ is NP-Complete.

In other words, even for this simplified version of the specification revision problem, it is unlikely that there exists a polynomial time algorithm that can solve the problem. The best we can hope for is a polynomial time approximation algorithm as the one that we have presented in [16] for the unweighted version of the problem. This is currently an ongoing research effort.

A. (Mixed)-Integer Linear Program Formulation for WMRP

We first present an Integer Linear Program (ILP) formulation that solves CMLP for $R = \emptyset$. Our goal is to compute the least number of labels to remove from the edges of $G_{\mathcal{A}}$ while the path has weight less than K. In the following all the variables are Boolean, $in(v) \subseteq E$ denotes the incoming edges to node $v \in V$ and $out(v) \subseteq E$ denotes the outgoing edges from a node $v \in V$. We use one Boolean variable x_π for each atomic proposition $\pi \in \Pi$ and one Boolean variable x_e to model each edge of the graph $G_{\mathcal{A}}$. Hence, we formulate the following ILP problem.

$$\min \sum\nolimits_{\pi \in \Pi} x_\pi \tag{3}$$

$$\text{s.t. } \forall v \in V \backslash (V_f \cup \{v_s\}) \; . \sum_{e_i \in in(v)} x_{e_i} = \sum_{e_o \in out(v)} x_{e_o} \tag{4}$$

$$\sum\nolimits_{e_o \in out(v_s)} x_{e_o} = 1 \tag{5}$$

$$\sum\nolimits_{v \in V_f} \sum\nolimits_{e \in in(v)} x_e = 1 \tag{6}$$

$$\forall e \in E \; . \; |L(e)| x_e \leq \sum\nolimits_{\pi \in L(e)} x_\pi \tag{7}$$

The cost function (3) minimizes the number of atomic propositions that must be enabled. Constraint (4) captures the requirement that the incoming "flow" to a node should be equal to the outgoing "flow". However, no loops through the initial node are allowed. Constraint (5) captures the fact that one outgoing edge from the source node should be on the path. Similarly, there should be one incoming edge enabled to a sink node (constraint (6)). Finally, inequality (7) imposes the constraint that if an edge e is part of the path, i.e., $x_e = 1$, then the atomic propositions on the edge must be removed.

When $R = \emptyset$, then the cost constraint, i.e., that the enabled path has total weight less than K, is simply

$$\sum\nolimits_{e \in E} W(e) x_e \leq K \tag{8}$$

The ILP above solves the CMLP, but not the WMRP. In order to compute the optimal solution for WMRP, we need to add the requirement for a loop from a sink node back to itself. In order to enable a loop, we need to create a copy of the graph for each sink node. The constraint (6) is relaxed to

at least one incoming node since a path might visit multiple sink nodes. For the same reason, we need

$$\forall v \in V_f \; . \; \sum\nolimits_{e_i \in in(v)} x_{e_i} \geq \sum\nolimits_{e_o \in out(v)} x_{e_o} \qquad (9)$$

Also, we must add the constraint that if an incoming edge to a sink node is part of the path, then an outgoing edge from the sink node in the copy of the graph may be part of the path as well. Let v' be the copy of a node $v \in V_f$, then

$$\sum\nolimits_{e \in in(v)} x_e \geq \sum\nolimits_{e \in out(v')} x_e \qquad (10)$$

Each copy of the graph has only one accepting node. Furthermore, we need to add the constraint that only one accepting node is visited in all copies of graphs. This increase in the complexity of the problem formulation is needed because an edge that has been enabled in the finite part of the path does not necessarily have to be part of the loop. Finally, constraint (8) must be replicated for the copies of the graph.

The above ILP formulation for WMRP might be too difficult to solve if there are many accepting nodes. Alternatively, we can solve a sequence of ILP problems one for each accepting node in V_f. In this case, we solve the optimization problem (3) under constraints (4)-(8) and (10) where now V_f is a singleton and constraints (4), (6)-(8) are added to the copy of the graph as well.

However, if the cost function $\mathcal{C}_2$ is considered, i.e., $R \neq \emptyset$, then the ILP formulation above is not sufficient. In this case, for each node v, we need a variable $d_v \in \mathbb{R}_{\geq 0}$ that keeps track of the cost of the path that arrives at node v. We replace constraint (8) with the following constraints, but only in the copy of the graph since the cost function ignores the transient behavior. For all $e = (v, v') \in E$, if $v \notin R$

$$-W_{ub}(1 - x_e) \leq d_{v'} - d_v - W(e) \leq W_{ub}(1 - x_e) \qquad (11)$$

and if $v \in R$

$$-W_{ub}(1 - x_e) \leq d_{v'} - W(e) \leq W_{ub}(1 - x_e) \qquad (12)$$

where W_{ub} is a large enough constant. In our implementation, it is set to $W_{ub} = C + \max_{e \in E} W(e)$. Furthermore, at least one cost reset node must be visited

$$\sum\nolimits_{v \in R} \sum\nolimits_{e \in in(v)} x_e \geq 1 \qquad (13)$$

The resulting optimization problem is a Mixed-Integer Linear Program (MILP).

Remark 2: If there is some ordering in the importance of achieving certain goals as expressed by the atomic propositions in the specification, then these could be incorporated in the cost function (3) as weights to the variables x_π.

B. Linear Program Relaxation for WMRP

A simple MILP solver in Matlab which uses a branch-and-bound algorithm cannot handle problems of the size of Example 1 (8484 variables, 8480 inequalities and 539 equalities). Therefore, we directly solve the Linear Program (LP) relaxation for WMRP in order to get an approximate solution. The LP relaxation is derived by simply replacing the constraints that x_e, $x_\pi \in \{0, 1\}$ with $0 \leq x_e \leq 1$ and

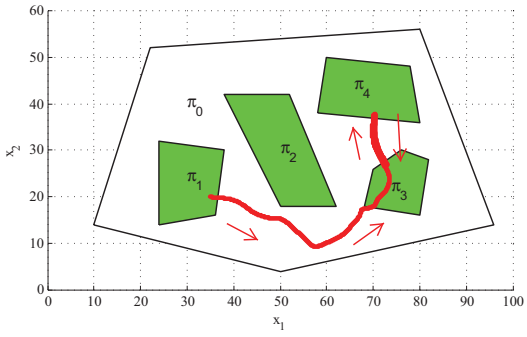

Fig. 2. Example 4: The resulting trajectory for the revision ϕ_{e2}.

$0 \leq x_\pi \leq 1$ for all edges e and atomic propositions π. If there is a feasible solution to the LP relaxation, we will get a vector of fractional numbers $x_{\pi_1}, \ldots, x_{\pi_n}$, where n is the number of atomic propositions in the specification.

However, the returned solution might be too conservative in the sense that most of the variables x_π could have some small non-zero value. Therefore, we employ 2 heuristics. First, we force the LP solver to look for feasible solutions around the minimal path found originally which does not satisfy the cost constraint. Second, we either randomly round the values of the variables x_π by assigning probabilities based on the value of the variables or we set a threshold, e.g., relative difference between variables, which depends on the problem. Finally, we also add the constraint $\sum_{\pi \in \Pi} x_\pi \geq 1$ since the variables can take very small values.

Example 4: The LP relaxation of WMRP using cost $\mathcal{C}_1$ on Example 1 is solved by Matlab linprog in less than 6sec on an i7 at 2.93GHz with 8GB. When we set as cost bound $C = 950$, we get as specification revision the set $\{\pi_2\}$. That is, the revised specification derived by replacing π_2 with $\top$ in ϕ_{e1} which is equivalent to $\phi_{e2} = \Box(\Diamond\pi_3 \wedge \Diamond\pi_4)$. The resulting trajectory appears in Fig. 2 with cost 855 for the prefix and 622 for the lasso part. The trajectory first visits π_3, then π_4 and then oscillates between π_3 and π_4.

However, when the cost bound is reduced to $C = 800$, then the revision returned is $\{\pi_2, \pi_3\}$. Thus, the equivalent LTL specification is $\phi_{e3} = \Box(\Diamond\pi_4)$, i.e., the robot arrives and stays in π_4. The resulting trajectory is presented in Fig. 3 with cost 597 for the prefix and 0 for the lasso part.

Under cost function $\mathcal{C}_1$, the LP relaxation is faster than exhaustive search.

Finally, if we set $\pi^\star = \pi_2$, which is allowed in our framework, and set the cost bound to 800, then the revision computed removes π_4. The cost of the resulting path, which repeatedly visits π_2 and π_3, is 722. However, this revision under cost function $\mathcal{C}_2$ took 19 min on the same computer. Future investigations will be focused on whether this is faster or slower than exhaustive search by repeatedly calling the optimal control algorithm from [17]. △

VI. Conclusions

Linear temporal logic can be viewed as a high level programming language for robots. In order for LTL control

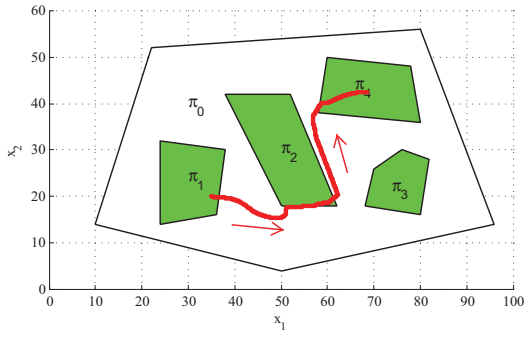

Fig. 3. Example 4: The resulting trajectory for the revision ϕ_{e3}.

methods to be adopted by a larger number of users, specification debugging and revision methods must be developed. In this paper, we introduced the problem of minimal revision of specification automata for linear temporal logic planning over weighted transition systems (WMRP). We proved that WMRP in a restricted form is NP-complete. For computing the exact solution to the problem, we provided a Mixed-Integer Linear Program (MILP) formulation of the WMRP. Since the low-end MILP solvers cannot solve realistic problem instances, we provided heuristics for a Linear Program (LP) relaxation albeit without any approximation guarantees. Future research will be targeted on developing heuristics and approximation algorithms with guaranteed bounds. We believe that a polynomial-time heuristic algorithm similar to [16] is necessary for implementing the algorithms to be run on embedded computing devices.

Acknowledgments: The authors would like to thank A. Agung Julius for recommending taking a linear programming approach. The authors would also like to thank the anonymous reviewers for their detailed comments and, in particular, reviewer 6 for suggesting considering the general case in specification (2).

REFERENCES

[1] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Translating structured english to robot controllers," *Advanced Robotics*, vol. 22, no. 12, pp. 1343–1359, 2008.

[2] J. Dzifcak, M. Scheutz, C. Baral, and P. Schermerhorn, "What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution," in *Proceedings of the IEEE international conference on robotics and automation*, 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, Feb. 2009.

[4] I. Filippidis, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Decentralized multi-agent control from local LTL specifications," in *51st IEEE Conference on Decision and Control*, 2012, pp. pp. 6235–6240.

[5] S. Karaman, R. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *IEEE Conf. on Decision and Control*, 2008.

[6] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic specifications," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 48–61, 2010.

[7] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *International Conference on Robotics and Automation*. IEEE, 2010, pp. 2689–2696.

[8] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon control for temporal logic specifications," in *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*. New York, NY, USA: ACM, 2010, pp. 101–110.

[9] P. Roy, P. Tabuada, and R. Majumdar, "Pessoa 2.0: a controller synthesis tool for cyber-physical systems," in *Proceedings of the 14th international conference on Hybrid systems: computation and control*, ser. HSCC '11. New York, NY, USA: ACM, 2011, pp. 315–316.

[10] L. Bobadilla, O. Sanchez, J. Czarnowski, K. Gossman, and S. LaValle, "Controlling wild bodies using linear temporal logic," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[11] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimal multi-robot path planning with temporal logic constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems,*, 2011, pp. 3087 –3092.

[12] B. Lacerda and P. Lima, "Designing petri net supervisors from ltl specifications," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[13] A. LaViers, M. Egerstedt, Y. Chen, and C. Belta, "Automatic generation of balletic motions," *IEEE/ACM International Conference on Cyber-Physical Systems*, vol. 0, pp. 13–21, 2011.

[14] G. E. Fainekos, "Revising temporal logic specifications for motion planning," in *Proceedings of the IEEE Conference on Robotics and Automation*, May 2011.

[15] K. Kim, G. Fainekos, and S. Sankaranarayanan, "On the revision problem of specification automata," in *Proceedings of the IEEE Conference on Robotics and Automation*, May 2012.

[16] K. Kim and G. Fainekos, "Approximate solutions for the minimal revision problem of specification automata," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012.

[17] S. L. Smith, J. Tumova, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *The International Journal of Robotics Research*, vol. 30, pp. 1695–1708, 2011.

[18] A. Cimatti, M. Roveri, V. Schuppan, and A. Tchaltsev, "Diagnostic information for realizability," in *Verification, Model Checking, and Abstract Interpretation*, ser. LNCS, F. Logozzo, D. Peled, and L. Zuck, Eds. Springer, 2008, vol. 4905, pp. 52–67.

[19] R. Konighofer, G. Hofferek, and R. Bloem, "Debugging formal specifications using simple counterstrategies," in *Formal Methods in Computer-Aided Design*. IEEE, Nov. 2009, pp. 152 –159.

[20] V. Raman and H. Kress-Gazit, "Analyzing unsynthesizable specifications for high-level robot behavior using LTLMoP," in *23rd International Conference on Computer Aided Verification*, ser. LNCS, vol. 6806. Springer, 2011, pp. 663–668.

[21] M. Göbelbecker, T. Keller, P. Eyerich, M. Brenner, and B. Nebel, "Coming up with good excuses: What to do when no plan can be found," in *Proceedings of the 20th International Conference on Automated Planning and Scheduling*. AAAI, 2010, pp. 81–88.

[22] D. E. Smith, "Choosing objectives in over-subscription planning," in *Proceedings of the 14th International Conference on Automated Planning and Scheduling*, 2004, p. 393401.

[23] M. van den Briel, R. Sanchez, M. B. Do, and S. Kambhampati, "Effective approaches for partial satisfaction (over-subscription) planning," in *Proceedings of the 19th national conference on Artifical intelligence*. AAAI Press, 2004, p. 562569.

[24] K. Sundar and S. Rathinam, "Route planning algorithms for unmanned aerial vehicles with refueling constraints," in *American Control Conference*, june 2012, pp. 3266 –3271.

[25] A. Pnueli, "The temporal logic of programs," in *Proceedings of the 18th IEEE Symposium Foundations of Computer Science*, 1977, pp. 46–57.

[26] J. R. Buchi, "Weak second order arithmetic and finite automata," *Zeitschrift für Math. Logik und Grundlagen Math.*, vol. 6, pp. 66–92, 1960.

[27] P. Wolper, "Constructing automata from temporal logic formulas: a tutorial," in *Lectures on formal methods and performance analysis: first EEF/Euro summer school on trends in computer science*. Springer, 2002, pp. 261–277.

[28] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. [Online]. Available: http://msl.cs.uiuc.edu/planning/

[29] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in *Proceedings of the 13th CAV*, ser. LNCS, G. Berry, H. Comon, and A. Finkel, Eds., vol. 2102. Springer, 2001, pp. 53–65.

[30] J. R. Buchi, "On a decision method in restricted second-order arithmetic," in *Proceedings of the 1960 International Congress on Logic, Methodology and Philosophy of Science*, 1962, pp. 1–11.

[31] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT Press/McGraw-Hill, Sep. 2001.