2016 IEEE International Conference on Robotics and Automation (ICRA)
Stockholm, Sweden, May 16-21, 2016

Dynamic Routing of Energy-Aware Vehicles with Temporal Logic Constraints

Derya Aksaray, Cristian-Ioan Vasile, and Calin Belta

***Abstract*— This paper addresses a persistent vehicle routing problem, where a team of vehicles is required to achieve a task repetitively. The task is given as a Time-Window Temporal Logic (TWTL) formula defined over the environment. The fuel consumption of each vehicle is explicitly captured as a stochastic model. As vehicles leave the mission area for refueling, the number of vehicles may not always be sufficient to achieve the task. We propose a decoupled and efficient control policy to achieve the task or its minimal relaxation. We quantify the temporal relaxation of a TWTL formula and present an algorithm to minimize it. The proposed policy has two layers: 1) each vehicle decides when to refuel based on its remaining fuel, 2) a central authority plans the joint trajectories of the available vehicles to achieve a minimally relaxed task. We demonstrate the proposed approach via simulations and experiments involving a team of quadrotors that conduct persistent surveillance.**

I. Introduction

In the Vehicle Routing Problem (VRP), the goal is to find N trajectories for N vehicles achieving a task (e.g., visiting all locations in minimum time). There are various extensions of VRP addressing time capacities, service time windows, service orders (e.g., [1], [2]), or uncertainty in service requests, travel time, or vehicle availability (e.g., [3], [4]). The VRPs are NP-hard combinatorial optimization problems. Typically, finding the optimal trajectories requires to explore all the possible routes. Such an exploration can be achieved by various optimization methods (e.g., integer linear programming, dynamic programming, branch and bound), whose computational complexities increase exponentially with the problem size. This has motivated the development of heuristics or approximate algorithms that result in acceptably good solutions with a lower complexity (e.g., [5]).

In some VRPs, simultaneous visits or relative timings between visiting particular locations might be critical for the task accomplishment. In general, if there exist some tasks that involve a temporal and logical ordering, it is hard to formulate them in the classical optimization setup. Temporal logics (TL) are rich and expressive specification languages that can be used to address this issue. For example, the authors of [6] and [7] address motion planning problems with specifications given in linear temporal logic. Alternatively, a VRP with metric temporal logic formulae is solved in [8].

This paper addresses a persistent VRP involving a group of energy-aware vehicles. The vehicles work together to satisfy a global task infinitely many times. The task is given as a Time-Window Temporal Logic (TWTL) formula over a set of locations. The semantics of TWTL is rich enough to capture a wide variety of timed temporal logic specifications, e.g., "Service A for 3 time units within [0, 5], and after this, service B for 2 time units within [4, 9]. Within 9 time units, if C is serviced for 2 time units, then D should be serviced for 3 time units." Each vehicle is assumed to have a stochastic fuel consumption model, and it leaves the mission area for refueling when necessary. We propose a decoupled and efficient control policy, in which each vehicle makes an individual decision for refueling whenever it reaches a critical fuel threshold, and a centralized controller plans only the trajectories of the vehicles in the mission area.

This paper is related to [2], where TWTL was first defined and used for persistent VRP with explicit time constraints. This work extends the results from [2] in three ways. First, the method in [2] is an off-line strategy, which can not handle uncertainty. In this paper, due to stochasticity in fuel consumption, we propose an *on-line* control policy that recomputes the trajectories during the mission whenever a change occurs in the number of available vehicles. Second, by decoupling refueling decisions from trajectory planning, the proposed policy exhibits a significantly lower computational complexity than the strategy presented in [2]. Third, while [2] returns failure in cases where the given TWTL formula cannot be satisfied, here we allow for satisfaction of minimally relaxed formulae. We introduce and quantify the *temporal relaxation* of a TWTL formula and compute trajectories by minimizing it. Accordingly, the resulting trajectories provide the best possible satisfaction performance.

Notation: $q^{[d]}$ denotes d repetitions of q. Given $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$, $n \geq 2$, the relationship $\mathbf{x} \sim \mathbf{x}'$, where $\sim \in \{<, \leq, >, \geq\}$, is true if it holds for all components. $\mathbf{x} \sim a$ denotes $\mathbf{x} \sim a\mathbf{1}_n$, where $a \in \mathbb{R}$ and $\mathbf{1}_n$ is the n-dimensional vector of all ones.

II. Time Window Temporal Logic (TWTL)

A. Syntax and Semantics

The syntax of TWTL is defined as:

$$\phi ::= \top \mid s \mid \phi_i \wedge \phi_j \mid \neg\phi_i \mid \phi_i \cdot \phi_j \mid H^d s \mid [\phi_i]^{[a,b]},$$

where $\top$ is the "true" constant; $s \in \mathcal{S}$ is a site label and $\mathcal{S}$ is a set of site labels; $\wedge$ and $\neg$ are the Boolean operators for conjunction and negation , respectively; $\cdot$ is the concatenation operator; H^d with $d \in \mathbb{Z}_{\geq 0}$ is the *hold* operator; and $[\ ]^{[a,b]}$ with $0 \leq a \leq b$ is the *within* operator. The semantics is defined with respect to finite output words $\mathbf{o}$ over $\mathcal{S}$. Let $\mathbf{o}(k)$ denote the k^{th} element on $\mathbf{o}$. The Boolean operators retain their usual semantics. The *hold* operator $H^d s$ specifies that a site $s \in \mathcal{S}$ should be serviced (satisfied) for d time

This work was partially supported by NSF NRI-1426907, NSF CMMI-1400167, ONR MURI N00014-09-1051 at Boston University. Derya Aksaray, Cristian-Ioan Vasile, and Calin Belta are with Boston University, MA, USA. {daksaray, cvasile, cbelta}@bu.edu

units (i.e., $\mathbf{o} \models H^d s$ if $\mathbf{o}(t) = s \ \forall t \in [0, d]$). The *within* operator $[\phi]^{[a,b]}$ bounds the satisfaction of ϕ with $[a, b]$ time window (i.e., $\mathbf{o} \models [\phi]^{[a,b]}$ if $\exists k \in (0, b-a)$ s.t. $\mathbf{o}' \models \phi$ where $\mathbf{o}' = \mathbf{o}(a+k) \dots \mathbf{o}(b)$). Lastly, the concatenation of ϕ_i and ϕ_j (i.e., $\phi_i \cdot \phi_j$) specifies that first ϕ_i must be satisfied and then immediately ϕ_j must be satisfied.

The satisfaction of a TWTL formula can be decided within bounded time. Accordingly, the *time bound* of ϕ, denoted as $\|\phi\|$, is the maximum time needed to satisfy ϕ, and it is computed as follows:

$$\|\phi\| = \begin{cases} 0 & \text{if } \phi \in \{\top, s\} \\ \max(\|\phi_1\|, \|\phi_2\|) & \text{if } \phi = \phi_1 \wedge \phi_2 \\ \|\phi_1\| & \text{if } \phi = \neg\phi_1 \\ \|\phi_1\| + \|\phi_2\| + 1 & \text{if } \phi = \phi_1 \cdot \phi_2 \\ d & \text{if } \phi = H^d s \\ b & \text{if } \phi = [\phi_1]^{[a,b]} \end{cases} \quad (1)$$

An example of TWTL formula and its time bound are: "service A for 3 time units within $[0, 5]$, and after this, service B for 2 time units within $[4, 9]$", $\phi = [H^3 A]^{[0,5]} \cdot [H^2 B]^{[4,9]}$, $\|\phi\| = 15$. The reader is referred to [9] for more details.

B. Temporal Relaxation

In this section, we introduce a novel notion called the *temporal relaxation* of a TWTL formula. This will be used in Sec. III to formulate the problem. To illustrate the main ideas, consider the following TWTL formula:

$$\phi_1 = [H^1 A]^{[0:2]} \cdot \left[H^3 B \wedge [H^2 C]^{[0:4]}\right]^{[1:8]}. \quad (2)$$

In cases where ϕ_1 cannot be satisfied, one question is: what is the "closest" achievable formula that can be performed? One way to do this is to relax the deadlines for the time windows, which are captured by the *within* operator. Accordingly, a relaxed version of ϕ_1 can be written as

$$\phi_1(\boldsymbol{\tau}) = [H^1 A]^{[0:(2+\tau_1)]} \cdot [H^3 B \wedge [H^2 C]^{[0:(4+\tau_2)]}]^{[1:(8+\tau_3)]}, \quad (3)$$

where $\boldsymbol{\tau} = (\tau_1, \tau_2, \tau_3) \in \mathbb{Z}^3$. Note that a critical aspect while relaxing the time windows is to preserve the feasibility of the formula. This means that any sub-task expressed as a sub-formula in ϕ should be achievable in the given time window.

Definition II.1 (Feasible TWTL formula)**.** *A TWTL formula ϕ is called feasible, if the time window corresponding to each* within *operator is greater than the duration of the corresponding enclosed task expressed via the* hold *operators.*

Remark II.1. *The formula $\phi_1(\tau)$ in* (3) *is a feasible TWTL formula, if each τ_i has the following constraint: (i) $2 + \tau_1 \geq 1$, (ii) $4 + \tau_2 \geq 2$, and (iii) $7 + \tau_3 \geq \max\{3, 4 + \tau_2\}$. Note that $\boldsymbol{\tau}$ may be non-positive. In such cases, $\phi_1(\boldsymbol{\tau})$ becomes a stronger specification than ϕ_1, which implies that the sub-tasks are performed earlier than their actual deadlines.*

Let ϕ be a TWTL formula. Then, a $\tau-$relaxation of ϕ is defined as follows:

Definition II.2 ($\tau-$Relaxation of ϕ)**.** *Let $\boldsymbol{\tau} \in \mathbb{Z}^m$, where m is the number of* within *operators contained in ϕ. The τ-relaxation of ϕ is a* feasible *TWTL formula $\phi(\boldsymbol{\tau})$, where each subformula of the form $[\phi_i]^{[a_i,b_i]}$ is replaced by $[\phi_i]^{[a_i,b_i+\tau_i]}$.*

Remark II.2. *For any ϕ, $\phi(\mathbf{0}) = \phi$.*

Remark II.3. *Let $\boldsymbol{\tau'}, \boldsymbol{\tau''} \in \mathbb{Z}^m$ such that $\phi(\boldsymbol{\tau'})$ and $\phi(\boldsymbol{\tau''})$ are feasible relaxations, where m is the number of* within *operators in ϕ. Note that if $\boldsymbol{\tau'} \leq \boldsymbol{\tau''}$, then $\phi(\boldsymbol{\tau'}) \Rightarrow \phi(\boldsymbol{\tau''})$.*

Definition II.3. *Given an output word $\mathbf{o}$, we say that $\mathbf{o}$ satisfies $\phi(\infty)$, i.e., $\mathbf{o} \models \phi(\infty)$, if and only if $\exists \boldsymbol{\tau'} < \infty$ s.t. $\mathbf{o} \models \phi(\boldsymbol{\tau'})$.*

Similar to Remark II.3, if $\boldsymbol{\tau} < \infty$, then $\phi(\boldsymbol{\tau}) \Rightarrow \phi(\infty), \forall\ \boldsymbol{\tau}$.

Proposition II.4. *Let $\phi(\boldsymbol{\tau'})$ and $\phi(\boldsymbol{\tau''})$ be two feasible relaxations. If $\boldsymbol{\tau'} \leq \boldsymbol{\tau''}$ then $\|\phi(\boldsymbol{\tau'})\| \leq \|\phi(\boldsymbol{\tau''})\|$.*

Definition II.4 (Temporal Relaxation)**.** *Given ϕ, let $\phi(\boldsymbol{\tau})$ be a feasible relaxed formula. The temporal relaxation of $\phi(\boldsymbol{\tau})$ is defined by $|\boldsymbol{\tau}|_{TR} = \max_j(\tau_j)$.*

Remark II.5. *If $|\boldsymbol{\tau}|_{TR} \leq 0$, then ϕ is satisfied.*

III. Problem Formulation

A. Environment Model

Consider an environment that contains a set of monitoring sites ($\mathcal{S}$) and a set of bases (or charging stations) ($\mathcal{C}$). Let $\mathcal{E} = (Q, \Delta, \varpi)$ denote a weighted directed connected graph, where $Q = \mathcal{S} \cup \mathcal{C}$ is the set of nodes representing the sites and the bases, $\Delta \subseteq Q \times Q$ is the set of edges representing the feasible travel between the nodes, and $\varpi : \Delta \to \mathbb{Z}_{\geq 1}$ is the edge weight that represents the travel time between the nodes. In this setting, we assume that there exists a path from any site to one of the bases without visiting any other sites (e.g., dashed edges in Fig. 1(a)).

B. Vehicle Model

Given $\mathcal{E} = (Q, \Delta, \varpi)$, a team of vehicles move on the edges Δ to pursue persistent operations. For any $q \in Q$, $\vec{q}$ denotes moving towards q. Let $\vec{Q} = \{\vec{q} | q \in Q\}$. At any t, the state of vehicle i is $[f_i(t), x_i(t)]$, where $f_i(t)$ is its remaining fuel, and $x_i(t) \in Q \cup \vec{Q}$ is its target state (i.e., either the node it is occupying or the node it is traveling to). In this paper, we only focus on the high-level planning. We assume that low-level controllers drive the vehicles from their current states to the designated target states (more information is provided in Sec. VI-B for a case when the vehicles are quadrotors).

Communication Model: We assume that 1) each vehicle can communicate with all the other vehicles through a complete communication graph, 2) there is no cost in communication, and 3) the information propagates significantly faster than the motion of the vehicles.

Fuel Model: Each vehicle has limited fuel capacity and consumes fuel unless it is located at a base. Accordingly, we use the following stochastic fuel model:

$$f(t+1) = \begin{cases} \min\left\{f(t) + \delta f^c, f_{max}\right\} & \text{if } \textit{at base}, \\ f(t) - \delta f^d + \xi(t) - \delta f^p & \text{otherwise}, \end{cases} \quad (4)$$

where $\delta f^c > 0$ is a constant refuel rate at the base, $\delta f^d > 0$ is a constant fuel consumption while operating, f_{max} is the maximum fuel capacity, $\xi(t)$ is a random variable modeling uncertainty in the fuel consumption, and $\delta f^p \in \{0, \beta_1, \beta_2\}$ models a fuel penalty if the vehicles avoid collisions through some maneuvers. In other words, if multiple vehicles travel

the same edge or operate at the same node, they avoid each other by modifying their trajectories, e.g., a change in flight altitude. Such operational changes typically cause more fuel consumption. Thus, $\delta f^p = \beta_1 > 0$ for each vehicle traveling the same edge or occupying the same node; $\delta f^p = \beta_2 > \beta_1$ for each vehicle traveling the same edge and arriving the same node simultaneously; $\delta f^p = 0$ in other cases.

C. Control Policy

In a persistent surveillance mission, each vehicle needs to (i) avoid running out of fuel, and (ii) work collaboratively to achieve a desired objective. Thus, each vehicle needs an efficient decision for when to refuel and how to move. In this paper, we propose to decouple the decision-making for refueling and operating in the surveillance area. In the proposed policy, each vehicle has a label as *active* or *inactive*. A vehicle changes its label from *active* to *inactive* if it decides to return to the base, whereas its label switches from *inactive* to *active* when it arrives at the surveillance area after refueling. We assume that each vehicle broadcasts any change in its label through the communication network. Then, a central authority assigns a target node to each active vehicle. Consequently, each vehicle's trajectory depends on two policies: the *refuel policy* results in a strategic decision for safe return to a base; the *operational control policy* results in efficient movement in the surveillance area.

1) Refuel Policy: In this paper, the vehicles follow a threshold policy for refueling. Accordingly, given a fuel threshold $f_i^{cr}(t)$ for an active vehicle i, if $f_i(t) > f_i^{cr}(t)$, then i remains to be active and it is in the surveillance area. Otherwise, i is inactive and moves towards a base.

2) Operational Control Policy: For M active vehicles, the operational control policy is a sequence $\mathbf{\Pi}_M = \pi(t)\pi(t+1)\dots$ where $\pi(t) \in (Q \cup \vec{Q})^M$ specifies at each time t and for each vehicle $i \in \{1, \dots, M\}$ where to stay or to go at $t+1$. We denote $\pi_i(t)$ as the target state of i at t and π_i as the control policy for i (the sequence of the target states).

D. Problem Definition

In this paper, achieving a persistent task means infinitely many satisfactions of a TWTL formula ϕ (i.e., $\mathbb{G}\phi$ where $\mathbb{G}$ stands for *always*). To formalize this concept, we define the infinite concatenation closure of ϕ as the concatenation of infinitely many copies of ϕ, i.e., $(\phi \cdot \phi \cdot \dots)$. Similarly, we define the infinite concatenation closure of relaxed TWTL formulae as $(\phi(\boldsymbol{\tau}^1) \cdot \phi(\boldsymbol{\tau}^2) \cdot \dots)$, where any $\phi(\boldsymbol{\tau}^i)$ corresponds to a $\boldsymbol{\tau}^i$-relaxation of ϕ. Note that a control policy $\mathbf{\Pi}_M = \pi(1)\pi(2)\dots$ induces an output word $\mathbf{o}$.

Definition III.1 (Output word)**.** *The output word generated by a control policy, $\mathbf{\Pi}_M = \pi(1)\pi(2)\dots$, is $\mathbf{o} = o_1 o_2 \dots$, where $o_t = \{\pi_i(t) | \pi_i(t) \in \mathcal{S}, i \in \{1, \dots, M\}\}$ is the set of all monitoring sites occupied by M vehicles at time t.*

Ideally, it is desired to find a policy that generates $\mathbf{o}$ satisfying $(\phi(\boldsymbol{\tau}^1) \cdot \phi(\boldsymbol{\tau}^2) \cdot \dots)$, where $\boldsymbol{\tau}^1 = \boldsymbol{\tau}^2 = \dots = \mathbf{0}$. However, $\boldsymbol{\tau}^i$ may contain nonzero elements due to uncertain vehicle availability in the surveillance area. In that case, the objective becomes to find a policy that minimizes $|\boldsymbol{\tau}^i|_{TR}$, i.e., the temporal relaxation.

Problem III.1. *Given an environment $\mathcal{E} = (\mathcal{S} \cup \mathcal{C}, \Delta, \varpi)$, M active vehicles, and a persistent task $\mathbb{G}\phi$, let Π_M generate an output word $\mathbf{o}$ that satisfies $(\phi(\boldsymbol{\tau}^1) \cdot \phi(\boldsymbol{\tau}^2) \cdot \dots)$. Find an optimal operational control policy*

$$\Pi_M^* = \arg\min_{\Pi_M} |\boldsymbol{\tau}^i|_{TR}\ , \quad \forall i. \tag{5}$$

Note that if M is constant during the mission, Π_M^* results in the optimal trajectories minimizing the temporal relaxation. However, M varies during the mission due to fuel uncertainty. Thus, solving (5) as M changes results in switching control policies. In Sec. V, we show that there always exists a solution under the switching policies and our proposed algorithm can find one. Nonetheless, resulting trajectories under the switching policies are not necessarily optimal.

IV. Control Synthesis

Our proposed solution to Prob. III.1 (Alg. 1) is inspired from automata-based model checking and has two phases. In the *off-line* computations, first, a list of active modes are created (e.g, a total number of N vehicles corresponds to N modes, where mode n represents the presence of n active vehicles in the environment). For each mode, a transition system is generated from the environment model. These are then combined with a special finite state automaton to obtain a list of product automata, each of which captures both motion and satisfaction in the corresponding mode. In the *on-line* computations, a centralized controller uses the product automata to compute the target states of all active vehicles. To this end, the control policy, Π_M, is computed on the currently active product automaton using a Dijkstra-based algorithm, and it is recomputed if any change occurs in M.

A. Multiple-Vehicle Motion

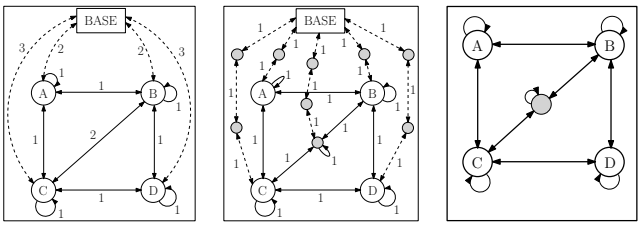
The motion model of a single vehicle is captured by a *deterministic transition system*.

Definition IV.1. *A* Deterministic Transtion System (DTS) *is a tuple $\mathcal{T} = (Q, q_0, \Delta, AP, h)$, where Q is the finite set of states; $q_0 \in Q$ is the initial state; $\Delta \subseteq Q \times Q$ is the set of transitions; AP is the set of observations; and $h : Q \to AP$ is the labeling function.*

The DTS of a vehicle is obtained by transforming the environment graph into an unweighted directed graph. To this end, we split up all transitions to have an edge weight of 1 and define new auxiliary states on the divided edges. Let $\mathcal{S}^{aux}$ and $\mathcal{C}^{aux}$ denote the set of auxiliary states between

Algorithm 1: Hybrid Control Policy

Input: $\mathcal{E} = (Q, \Delta, \varpi)$ environment, ϕ TWTL formula, N number of vehicles
Extract $\mathcal{T}$ and $\mathcal{T}^{full}$ from $\mathcal{E}$ and construct $\mathcal{T}^k$, $1 \le k \le N$ `// Off-line`
$\mathcal{A}_\infty \leftarrow translate(\phi)$, the FSA corresponding to $\phi(\infty)$
Create product automata $\mathcal{P}^k = \mathcal{T}^k \times \mathcal{A}_\infty$, $1 \le k \le N$
while *True* **do** `// On-line`
  foreach *active vehicle* **do**
    if *vehicle.fuel is critical* **then**
      vehicle.mode ← inactive; controls.inactive ← returnToBase()
  controls.active ← operationalPolicy(*active vehicles*)
  foreach *vehicle* **do**
    vehicle.move(); vehicle.updateFuel()
    if *vehicle.state* $\in \hat{Q}$ *(surveillance area)* **then** vehicle.mode ← active
    else vehicle.mode ← inactive



(a) Environment $\mathcal{E}$ (b) full DTS $\mathcal{T}^{full}$ (c) reduced DTS $\mathcal{T}$

Fig. 1. (a) Environment $\mathcal{E}$ containing four monitoring sites A, B, C, D and a base, (b) the full motion DTS on $\mathcal{E}$, and (c) the reduced DTS modeling motion only in the surveillance area.

the sites and between the sites and the bases, respectively. The DTS of a vehicle is $\mathcal{T}^{full} = (Q^{full}, q_0, \Delta^{full}, AP, h)$, where $Q^{full} = \mathcal{S} \cup \mathcal{C} \cup Q^{aux}$ and $Q^{aux} = \mathcal{S}^{aux} \cup \mathcal{C}^{aux}$; $q_0 \in Q^{full}$; $\Delta^{full} \subseteq Q^{full} \times Q^{full}$; $AP = \mathcal{S} \cup \{\epsilon\}$ where ϵ indicates that no site is occupied; and $h : Q^{full} \to AP$ is the labeling function such that it assigns a label to each site, i.e., $h(q) = q$ if $q \in \mathcal{S}$ and $h(q) = \epsilon$ for $q \notin \mathcal{S}$.

For example, Fig. 1(b) illustrates the *full DTS* of a vehicle, which is extracted from the environment in Fig. 1(a). We also define a *reduced DTS* as $\mathcal{T} = (Q^{red}, q_0, \Delta^{red}, AP, h)$, where $Q^{red} = \mathcal{S} \cup \mathcal{S}^{aux}$ involves only the sites and the auxiliary states connecting them as shown in Fig. 1(c).

We use $\mathcal{T}$ for the planning of active vehicles in the surveillance area. This enables to decouple the operational planning from the refuel decisions. Moreover, using a reduced DTS in planning significantly reduces the state-space of the overall system since the concurrent motion of the vehicles is represented by a *product transition system*.

Definition IV.2. *A* Product Transition System (PTS) $\mathcal{T}^k$ *for* $k \geq 1$ *is a DTS* $\mathcal{T}^k = (Q^k, q_0^k, \Delta^k, 2^{AP}, h^k)$*, where* $Q^k \subseteq Q^{red} \times \cdots \times Q^{red}$ *for* k *times is the set of states;* $q_0^k \in Q^k$ *is the initial state;* $\Delta^k \subseteq Q^k \times Q^k$ *is the set of transitions such that* $([x_1, \ldots x_k], [x'_1, \ldots x'_k]) \in \Delta^k$ *if* $(x_i, x'_i) \in \Delta^{red}$ *for all* $i \in \{1, \ldots, k\}$*;* 2^{AP} *is the set of observations (power set of AP); and* $h^k([x_1, \ldots, x_k]) = \{h(x_i) | i \in \{1, \ldots, k\}\}$.

In the proposed approach, the centralized controller only tracks the occupied states of $\mathcal{T}$ with multiplicities. Thus, we use quotient PTSs, whose states are equivalence classes induced by the permutation of the state vectors (e.g., the states (A, A, B), (A, B, A) or (B, A, A), representing 3 vehicles occupying A and B, are merged into a single state). This representation greatly reduces the sizes of the resulting PTSs, and any PTS along the paper implies a quotient PTS.

B. Specification

The specification is enforced using a *deterministic finite state automaton*.

Definition IV.3. *A* deterministic Finite State Automaton (FSA) *is a tuple* $\mathcal{A} = (S_\mathcal{A}, s_0, \Sigma = 2^{AP}, \delta, F_\mathcal{A})$*, where* $S_\mathcal{A}$ *is a finite set of states;* $s_0 \in S_\mathcal{A}$ *is the initial state;* Σ *is the input alphabet;* $\delta : S_\mathcal{A} \times \Sigma \to S_\mathcal{A}$ *is the transition function; and* $F_\mathcal{A} \subseteq S_\mathcal{A}$ *is the set of accepting states.*

Note that $\mathcal{A}$ can be constructed from any ϕ. However, $\mathcal{A}$ represents only the specification with the given time windows. In order to compactly represent all temporal relaxations of ϕ, a special automaton $\mathcal{A}_\infty$ is constructed based on the procedure in [9]. Accordingly, $\mathcal{A}_\infty$ represents $\phi(\boldsymbol{\tau})$ for all possible $\boldsymbol{\tau}$.

C. Operational Control Policy

The operational control policy is computed on the product automata between the PTSs and $\mathcal{A}_\infty$, which capture both the motion of the active vehicles and satisfaction of the formula.

Definition IV.4. *A* Product Automaton (PA) $\mathcal{P}_k = \mathcal{T}^k \times \mathcal{A}_\infty$ *for* $1 \leq k \leq N$ *is a tuple* $\mathcal{P}_k = (S_{\mathcal{P}_k}, (q_0^k, s_0), \Delta_{\mathcal{P}_k}, F_{\mathcal{P}_k})$*, where* $S_{\mathcal{P}_k} = Q^k \times S_{\mathcal{A}_\infty}$ *is the finite set of states;* $(q_0^k, s_0) \in S_{\mathcal{P}_k}$ *is the initial state;* $\Delta_{\mathcal{P}_k} \subseteq S_{\mathcal{P}_k} \times S_{\mathcal{P}_k}$ *is the set of transitions;* $F_{\mathcal{P}_k} = Q^k \times F_{\mathcal{A}_\infty}$ *is the set of accepting states.*

A transition $\big((q, s), (q', s')\big) \in \Delta_{\mathcal{P}_k}$ implies $(q, q') \in Q^k$ and $s \overset{h(q)}{\to}_{\mathcal{A}_\infty} s'$. The notions of trajectory and acceptance are the same as in FSA. A satisfying run of $\mathcal{T}^k$ with respect to ϕ can be obtained by computing a path from the initial state to an accepting state over $\mathcal{P}_k$ and projecting the path onto $\mathcal{T}^k$.

We propose Alg. 2 (line 8 in Alg. 1) to compute the target states of the active vehicles at each time step. Alg. 2 stores a local policy generated on the currently selected PA $\mathcal{P}$ and the last returned PA state $p = (q, s) \in S_\mathcal{P}$, where q is the PTS state, and s is the state on $\mathcal{A}_\infty$. The switching between PAs occurs when the last stored q is different than the actual PTS state of active vehicles q' (line 4). When s reaches an accepting state and the policy becomes empty, s is set to the initial state of $\mathcal{A}_\infty$ and the next satisfaction of ϕ initiates (line 8). Using $\mathcal{P}$ and p, the target states of the active vehicles are computed in line 10 by $computePolicy()$, which proceeds by traversing the structure of ϕ from smaller to larger sub-formulae. It uses special annotation on the automaton $\mathcal{A}_\infty$ to compute satisfying paths in $\mathcal{P}$ without considering within operators. Then, these paths are recursively filtered and extended based on the boolean and temporal operators connecting them. If there is no satisfying policy, then the procedure returns the current p. The detailed description of $computePolicy()$ can be found in [9]. The target states are distributed to vehicles via Hopcroft-Karp algorithm (line 11).

V. Analysis of the Hybrid Control Policy

A. Performance

First, we show that a relaxed TWTL formula can always be satisfied under an assumption on vehicle capabilities.

Definition V.1 (Operational Cycle)**.** *An operational cycle of a vehicle is an ordered sequence of traveling to the area, operating in the area, returning to the base, and refueling.*

Algorithm 2: On-line planning – $operationalPolicy()$

Input: the set of active vehicles; **Output:** the next state for each active vehicle
Data: $p = (q, s)$ – last PA state,$\mathcal{P}$ – selected PA, $policy$ – current policy

1 **if** *no active vehicles* **then return** $\{\}$
2 $q' = [vehicle.state \mid vehicle.mode = active]$
3 $replan = False$
4 **if** $q \neq q'$ **then** // any change in the PTS state
5 $\mathcal{P} \leftarrow \mathcal{P}_{|q'|}$; $q \leftarrow q'$; $replan \leftarrow True$ // switch PA
6 **else**
7 **if** $policy \neq \{\}$ **then** $p = policy.next()$ **else** $replan \leftarrow True$
8 **if** $s \in F_{\mathcal{A}_\infty}$ **then** $s \leftarrow s_0$ // update FSA state
9 **if** $replan = True$ **then**
10 $policy \leftarrow computePolicy(\mathcal{P}, p)$; $p \leftarrow policy.next()$
11 **return** $distributeControls(q)$

Definition V.2. *The* Abstract Syntax Tree *of ϕ is denoted by $AST(\phi)$, where each leaf is an atomic proposition $s \in \mathcal{S}$ or $\top$, and each intermediate node corresponds to an operator.*

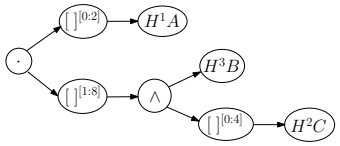


Fig. 2. The AST corresponding to $[H^1A]^{[0:2]} \cdot \left[H^3B \wedge [H^2C]^{[0:4]}\right]^{[1:8]}$.

Definition V.3 (Formula Primitive)**.** *Given ϕ, a formula primitive is a maximal subtree in $AST(\phi)$, which does not contain a within operator.*

Assumption 1. *Given ϕ, a vehicle is able to satisfy any ϕ primitive(s) at least once in one operational cycle.*

Consider $\phi = [H^4A]^{[3,8]} \wedge [H^2B \cdot H^1C]^{[4,9]}$ whose formula primitives are H^4A and $H^2B \cdot H^1C$. Assump. 1 implies that, in one operational cycle, the vehicle can reach A, stay there for 4 time steps, and return to the base safely. Similarly, it can also reach B, stay there for 2 time steps, then reach C, stay there for 1 time step, and return to the base.

Definition V.4 (Feasible Sequence of Formula Primitives)**.** *Given ϕ, a feasible sequence of formula primitives is an ordered sequence of formula primitives, whose overall satisfaction implies a feasible relaxation of ϕ.*

Again, consider $\phi = [H^4A]^{[3,8]} \wedge [H^2B \cdot H^1C]^{[4,9]}$. There exist only two feasible sequences of formula primitives, which are $(H^4A, H^2B \cdot H^1C)$ and $(H^2B \cdot H^1C, H^4A)$.

Theorem V.1. *Let ϕ be a TWTL formula. If Assump. 1 holds, then there always exists a* feasible sequence *that induces a valid relaxation $\phi(\boldsymbol{\tau})$ such that $\left\|\phi(\boldsymbol{\tau}^i)\right\| \leq k\, t^*_{OC}$, where k is the length of the longest feasible sequence of ϕ primitives, and t^*_{OC} is the maximum duration to finish a cycle.*

B. Safety

In Alg. 1, the decision to refuel (be inactive) based on the threshold policy should ensure safe return to the base.

Proposition V.2. *Let $\mathcal{N}_i(t)$ be the set of adjacent nodes to vehicle i on $\mathcal{T}^{full}$ and let $\bar{\xi}$ be the maximum uncertainty in the fuel consumption. Executing Alg. 1 ensures safe return to the base for vehicle i, if the refuel policy has a threshold*

$$f_i^{cr}(t) \geq \max_{q_j \in \mathcal{N}_i(t)} (1 + \varpi_{q_j q_B})(\delta f^d + \bar{\xi} + 2\beta) \qquad (6)$$

C. Complexity

In Alg. 1, the construction of $\mathcal{T}$ and $\mathcal{T}^{full}$ has complexity $O(\sum_e \varpi_e)$, where ϖ_e is the weight of edge e in $\mathcal{E}$. For N vehicles, the complexity of constructing all PTS is $O\left(\binom{|Q^{red}|+N}{N}\right)$ since the size of PTS $\mathcal{T}^k$ is equal to the number of permutations of k objects from Q^{red} with repetitions. Constructing $\mathcal{A}_\infty$ from ϕ has complexity $O(2^{|\phi|})$ where $|\phi|$ is the length of the formula [9]. Finally, the complexity of computing each PA $\mathcal{P}_k$ is $O(|Q^k| \cdot |S_{\mathcal{A}_\infty}|)$. In Alg. 2, the on-line planning is $O(|S_{\mathcal{P}_k}| + |\Delta_{\mathcal{P}_k}|)$ for $1 \leq k \leq N$ [9]. Moreover, how to return to the base is computed by running Dijkstra's algorithm on $\mathcal{T}^{full}$, which gives a complexity of $O(|Q^{full}| + |\Delta^{full}|)$. Overall, we improve the complexity of the solution (compared to the one in [2]) by 1) using multiple smaller PAs instead of a single complex PA, 2) decoupling the refuel decision from the trajectory planning, which significantly reduces the state-space, 3) representing ϕ via the special automaton $\mathcal{A}_\infty$ in a more compact way, and 4) using quotient PTSs instead of the normal ones.

VI. Case Study

A. Simulation Results

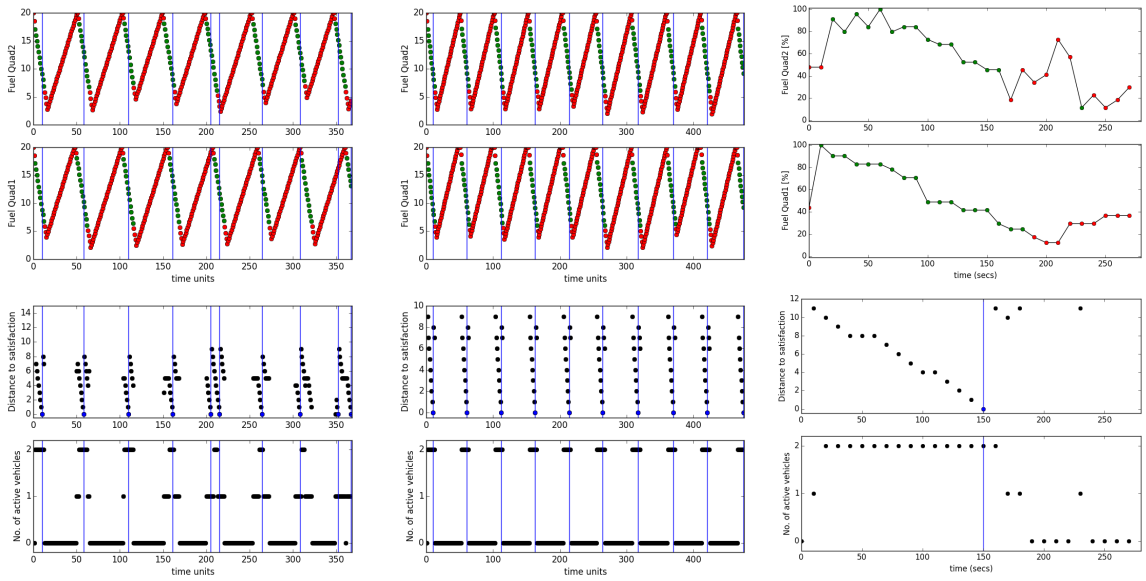
We consider two identical vehicles, an environment with four sites and a base as in Fig. 1(a), and the TWTL formula $\phi = [H^2A]^{[0,8]} \cdot [H^3B \wedge [H^2C]^{[1,5]}]^{[0,7]} \cdot [H^1D]^{[0,3]}$, which means "perform in order: 1) service A for 2 time units within $[0, 8]$; 2) within $[0, 7]$, service B for 3 time units and service C for 2 time units within $[1, 5]$; and 3) service D for 1 time unit within $[0, 3]$". Note that $\|\phi\| = 20$ so a single satisfaction of ϕ needs to be achieved in 20 time units. The parameters of each vehicle are selected as $f_{max} = 20$, $\delta f^c = 0.5$, $\delta f^d = 1$, $\delta f^p \in \{0, 0.2, 0.4\}$, $\xi(t) \sim \text{unif}(-0.1, 0.1)$.

The simulations were implemented in Python2.7 on a Intel Core i7 laptop with a 1.8 GHz processor and 8GB memory. ϕ was translated to $\mathcal{A}_\infty$ with 16 states and 36 transitions in 9 msec. The construction of $\mathcal{T}^1$, $\mathcal{T}^2$, $\mathcal{P}_1$, and $\mathcal{P}_2$ took <1 msec, 1 msec, 3 msec, and 19 msec, respectively. Moreover, $\mathcal{T}^1$, $\mathcal{T}^2$, $\mathcal{P}_1$, and $\mathcal{P}_2$ have 5, 15, 80, 240 states and 11, 78, 255, 2115 transitions, respectively. Overall, the off-line computation of Alg. 1 took 65 msec, while a single iteration in its on-line computation took less than 1 msec.

The remaining fuel of each vehicle at each time step is displayed in Fig. 3(a). The green and red markers indicate that the vehicle is *active* and *inactive*, respectively. In order to measure the progress towards satisfaction, we define the *distance to satisfaction* (d_{sat}) at each t as the length of the shortest path in $\mathcal{A}_\infty$ from the current specification state to a final state. The vertical lines in this figure indicate a single satisfaction of $\phi(\boldsymbol{\tau}^i)$, which we call a satisfaction loop. The first 2 relaxed formulae satisfied by the vehicles are: $\phi(\boldsymbol{\tau}^1) = [H^2A]^{[0,8-\mathbf{4}]} \cdot [H^3B \wedge [H^2C]^{[1,5-\mathbf{2}]}]^{[0,7-\mathbf{4}]} \cdot [H^1D]^{[0,3-\mathbf{2}]}$; $\phi(\boldsymbol{\tau}^2) = [H^2A]^{[0,8-\mathbf{6}]} \cdot [H^3B \wedge [H^2C]^{[1,5+\mathbf{37}]}]^{[0,7+\mathbf{35}]} \cdot [H^1D]^{[0,3-\mathbf{2}]}$.

In Fig. 3(a), d_{sat} decreases if there is at least one active vehicle. While two vehicles are active, whenever one of them becomes inactive, d_{sat} increases abruptly. Also, if there are no active vehicles, d_{sat} becomes undefined, shown as gaps in Fig. 3(a). If both vehicles return to the base and $d_{sat} \neq 0$, the satisfaction loop is not re-initiated. Instead, whenever a vehicle becomes active, it continues to make progress for the uncompleted loop. Hence, the results demonstrate that a relaxed ϕ is eventually satisfied in a periodic fashion.

We also compare the proposed policy with a benchmark policy (Π_B) where a relaxation is not allowed. In other words, if ϕ can not be satisfied by the active vehicles, all vehicles return to the base. While Π_B results in only the satisfaction of ϕ, it causes a significant amount of time gaps between the satisfactions as illustrated in Fig. 3(b). Note that 10 satisfactions of ϕ require 550 time steps whereas 10 satisfactions of the relaxed formulae are achieved in 380 time steps. The results indicate that allowing temporal relaxation of a formula increases the number of satisfactions.



(a) (b) (c)

Fig. 3. (a) Simulation results: 10 satisfactions of the relaxed formulas via the proposed policy, (b) Simulation results: 10 satisfactions of the original formula via the benchmark policy, (c) Experimental results with two quadrotors.

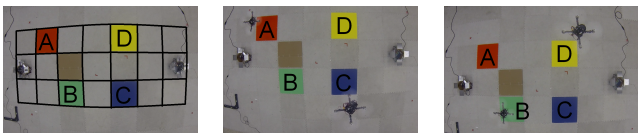
B. Experimental Results

We present some preliminary results on a multi-quadrotor testbed at the BU Robotics Laboratory. The flight space is equipped with an indoor OptiTrack localization system, which tracks reflective markers mounted on K500 quadrotors from KMel Robotics. Each quadrotor is equipped with an 11.57 V 3-cell LiPo battery and custom charging gear, which allows them to automatically recharge their batteries at a charging station. The quadrotors hover and move via local controllers, which were designed based on the differential flatness property of the quadrotors' dynamics [10].

We consider two quadrotors and a grid environment with 4 sites and 2 charging stations as in Fig. 4(a). A quadrotor can move to any adjacent cell other than the brown cell (representing an obstacle). A unique flight altitude and charging station is assigned to each quadrotor to avoid collisions. The objective is to satisfy $\phi = [H^2 A \wedge H^2 C]^{[0,8]} \cdot [H^3 B \wedge H^3 D]^{[0,7]} \cdot [[H^2 A]^{[2,6]} \vee [H^2 C]^{[1,5]}]$ repeatedly. The remaining fuel, the distance to satisfaction, and the number of active vehicles are shown in Fig. 3(c) . Fuel in this case is interpreted as battery voltage level. In Fig. 3(c), there exists some fluctuations in the remaining fuel due to the potential measurement errors, but a decreasing trend is observed in both the remaining fuel and the distance to satisfaction.

VII. Conclusion

We considered a persistent vehicle routing problem involving a team of vehicles that are required to achieve a task repetitively while refueling when necessary. We expressed the task as a TWTL formula over a set of locations. We proposed a hybrid control policy that decouples the refueling decision of each vehicle from the joint planning in the mission area. The proposed policy has two main benefits. First, the trajectories are computed on-line, and they are updated whenever a change occurs in the mission area. Second, if the TWTL formula is unsatisfiable, the trajectories for the active vehicles are computed by minimally relaxing the formula. To achieve this, we introduced a new notion called "temporal relaxation". We demonstrated the performance of the proposed policy via simulations and experiments.



(a) Initial state (b) Servicing A and C (c) Servicing B and D

Fig. 4. 2 quadrotors in an environment with 4 sites and 2 charging stations.

References

[1] P. Toth and D. Vigo, Eds., *The Vehicle Routing Problem*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001.

[2] C. Vasile and C. Belta, "An Automata-Theoretic Approach to the Vehicle Routing Problem," in *Proc. of the Robotics: Science and Systems Conference (RSS)*, Berkeley, California, USA, July 2014.

[3] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, "Dynamic vehicle routing for robotic systems," *Proc. of the IEEE*, vol. 99, no. 9, pp. 1482–1504, 2011.

[4] Q. Mu, Z. Fu, J. Lysgaard, and R. Eglese, "Disruption management of the vehicle routing problem with vehicle breakdown," *Journal of the Operational Research Society*, vol. 62, no. 4, pp. 742–749, 2011.

[5] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 3, pp. 345–358, 1992.

[6] A. Bhatia, L. Kavraki, and M. Vardi, "Sampling-based motion planning with temporal goals," in *IEEE Int. Conference on Robotics and Automation (ICRA)*, 2010, pp. 2689–2696.

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

[8] S. Karaman and E. Frazzoli, "Vehicle routing problem with metric temporal logic specifications," in *IEEE Conference on Decision and Control*, 2008, pp. 3953 – 3958.

[9] C.-I. Vasile, D. Aksaray, and C. Belta, "Time Window Temporal Logic," *arXiv preprint arXiv:1602.04294v1*, 2016.

[10] K. Leahy, D. Zhou, C.-I. Vasile, K. Oikonomopoulos, M. Schwager, and C. Belta, "Provably correct persistent surveillance for unmanned aerial vehicles subject to charging constraints," in *Experimental Robotics*. Springer, 2016, pp. 605–619.