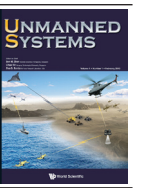
Unmanned Systems, Vol. 5, No. 4 (2017) 223–236

DOI: 10.1142/S2301385017500108

UNMANNED SYSTEMS
http://www.worldscientific.com/worldscinet/us



Unpredictably Dynamic Environment Patrolling

Jinwoo Seok*,‡, Mariam Faied†, Anouck Girard*

*Department of Aerospace Engineering
University of Michigan, 1320 Beal Ave, Ann Arbor, MI 48109, USA

†Department of Electrical and Computer Engineering
University of Detroit Mercy, 4001 W McNichols Rd, Detroit, MI 48221, USA

Patrolling is the most commonly requested type of unmanned aerial vehicle (UAV) mission; thus we use it as a prototypical example of autonomous mission planning. In this paper, we design an autonomous mission planner for UAV patrolling missions in unpredictably dynamic environments (UDEs) using recomposable restricted finite state machines (ReRFSM). UDEs admit online changes to the number of waypoints, their priorities, and the availability of paths without any prediction. ReRFSM can model UDEs and restricted inputs. Dynamic programming is used to generate policies and limited breadth-first search (LBFS) is used to generate solutions with a small computation time that plan single UAV tours. Finally, we introduce patrolling policies for multiple UAVs by composing multiple mission RFSMs and modifying the inputs of the ReRFSMs. We demonstrate the case of two vehicles as a proof of concept.

Keywords: UAV; patrolling; dynamic environments.

US

1. Introduction

1.1. *Motivation*

Unmanned Aerial Vehicles (UAVs) are a key technology for the United States Air Force in the coming decades. Their use has been growing rapidly, for a greater variety of missions, due to increased autonomous decision making capabilities, including mission planning. Autonomous decision making works best if it accounts for the local environment as well as for whether the environment is allowed to change [1]. Patrolling is the most commonly requested type of UAV mission; thus we use it as a prototypical example of autonomous mission planning. Patrolling missions are used both for civilian and military applications. Examples of civilian applications include search for survivors of accidents or natural disasters, and surveillance for crime prevention. Examples of military applications include searching for objects or facilities, reconnaissance of contested areas, surveillance of important facilities and/or base camps, and data collection from unattended sensors.

In patrolling, UAVs visit areas of interest to collect information autonomously. To maximize the patrolling performance of the UAVs given the area of interest, tours for the available UAVs that account for the mission environment need to be found. Otherwise, the UAVs may spend more fuel, time or may fail to accomplish the mission. Thus, to determine the best UAV tours for the patrolling mission, parameters of the mission environment have to be considered, such as the size and number of areas of interest, the kinds of tasks that need to be accomplished in each area of interest, and flight path conditions between the areas of interest.

However, the world is unpredictable, especially in adversarial situations, and thus the mission environment may change unpredictably in time. As such, which areas are of interest may change: a UAV may no longer have to visit areas it was previously assigned to but may have to visit new areas of interest instead. Furthermore, the risk in some areas of interest may evolve in time; previously safe areas may become too dangerous to fly over. These changes can be caused by the information collected during the mission,

Received 16 November 2016; Revised 11 August 2017; Accepted 12 August 2017; Published 10 October 2017. This paper was recommended for publication in its revised form by editorial board member, Yunfeng Zhang.
Email Address: ‡*sjinu@umich.edu*

so it may not be possible to know about the changes in advance. In short, we allow for Unpredictably Dynamic Environments (UDEs). If a UAV has a mission plan that only considers the initial environment, it may not be able to accomplish the patrolling mission. Therefore, the United States Air Force and others strive for autonomous mission planning in UDE, as described in [1].

Consider the following UAV patrolling mission scenario: A UAV flies over a contested area to gather information on objects of interest. The objects of interest may be targets, important facilities or likely hostage locations. There are initial waypoints of interest to search and each waypoint has its own priority, hence the UAV may need to visit the higher priority waypoints as soon as possible. Furthermore, some of the paths between the waypoints may not be available because of dangerous flight conditions such as enemy presence or weather. Once the initial tour is decided, the UAV starts to gather information. Based on the collected information, some waypoints might not warrant a visit anymore and new waypoints might be added. In addition, some safe paths may in truth be unsafe, or some unsafe paths may be safe. Finally, the UAV's patrol may be interrupted for many reasons, such as a human operator wanting to check a specific waypoint immediately or some waypoints being impossible to visit (e.g. it may be impossible to get a clear picture).

Thus, we develop a UAV mission planner that can handle such UDE patrolling scenarios. We design a mission planner for the UAV patrolling problem in UDE using Recomposable Restricted Finite State Machines (ReRFSM). ReRFSM can handle UDE by allowing the state space of a Finite State Machine (FSM) to change dynamically to follow the environment. ReRFSM can represent the problem by prohibiting some of the transitions during the composition. By applying Dynamic Programming (DP) to ReRFSM, a policy is obtained and by applying a heuristic method, Limited Breadth-First Search (LBFS), to ReRFSM, a solution is obtained. By solution, we mean the optimal/suboptimal sequence of decisions at the current state. By policy, we mean the solutions for all the states. DP yields an optimal and robust policy. By robust, we mean that DP yields a policy for every state, so even though a state may change unexpectedly, we always have a policy for the resulting state. Thus, even though the current plan may be interrupted, the policy has an alternative plan within the state space. This comes at the cost of computation; the method is thus applicable to small problem instances. For large problem instances, LBFS quickly generates a solution for the current state.

1.2. *Literature review*

Autonomous vehicle patrolling has been studied at great length in the literature. In [2], the problem is modeled using a graph, and each vertex has a relative deadline, which is the time for the intruders to reach a base from the vertex. Thus, the UAV has to visit all the vertices in such a way that the intruders do not have enough time to reach the base from their current vertex. UAV surveillance for Unattended Ground Sensors (UGSs) has been studied in [3, 4]. In [4], teamed UAVs perform surveillance near a base relying on UGSs placed on the roads to detect the intruders. The UAVs have to visit UGSs to find intruders, so the path is determined based on revisit deadlines of UGSs and the probability of intercepting the intruder. In [5], a scheme is presented for dynamic classification-driven sensor optimization under incomplete information and for objects with time-varying dynamics.

The UAV patrolling mission treated in this paper is similar to the Traveling Salesman Problem (TSP) or Vehicle Routing Problem (VRP). The TSP is formulated as follows: given the locations of a set of cities, find the shortest tour for a salesman such that the salesman visits all the cities exactly once. There exist many variations and heuristic solutions of the TSP in the literature. One of the most successful heuristics is the k-opt heuristic, specifically the LK heuristic [6]. An algorithm that transforms the heterogeneous, multiple depot, multiple UAVs routing problem to the Asymmetric TSP (ATSP) is proposed in [7], so the well known LK heuristic can be applied to solve the problem. In [8], the authors solve the TSP for Dubins' vehicles, that are a standard kinematic model abstraction for UAVs, where the vehicles are assumed to operate in a 2D plane and have turn rate constraints, and the paths between points may be straight lines or arcs of circles.

The VRP originated in a truck dispatching problem in [9]. The truck dispatching problem is a generalized version of the TSP: Given fleets of delivery trucks, bulk terminals, and many service locations to visit, find a route for each truck such that all service locations are visited and the total length of all routes is minimized, considering that the trucks have limited capacity. The VRP is formulated in more detail in [10]; the general form of the problem under consideration is: Given a set of locations with requirements and limited capacity vehicles that deliver goods from a single depot, minimize the total cost. Many exact solutions and approximation algorithms for the VRP have been proposed; a number are discussed in [11]. Minimizing emissions and fuel consumption are considered in [12].

Dynamic versions of the VRP are proposed in [13] (the Dynamic Vehicle Routing Problem (DVRP)) and in [14] (the Dynamic Traveling Repairman Problem (DTRP)). They consider dynamically changing environments. In [15], the UAV is required to visit a dynamically growing set of targets while traveling on a Dubins path. The authors propose an algorithm to find solutions and study the stability of solutions to the problem. This work is extended in [16] for multiple UAVs. In [17], the authors use a queueing approach

to design cooperative control and task allocation strategies for DVRP with priorities, and many variations of VRP exist in the literature [18–20].

The literature does not contain methods that allow one to deal with simultaneous addition and deletion of waypoints, (changing) priorities of waypoints and (changing) availability of paths between waypoints, for example for UAV patrolling. These scenarios are, however, highly realistic. Thus, in this paper, we aim to develop a UAV mission planner that can handle such an UDE and consider the effects of priorities of the waypoints and no fly zones on the paths.

1.3. *Original contributions*

The original contributions of this paper are as follows:

1. We design UDE patrolling strategies in the FSM framework using ReRFSM. UDE allows for changing online the number of waypoints, their priorities, and availabilities of the paths without any predictions. Our ReRFSM design can handle UDE by allowing the state space to change vigorously to accommodate the changing environment without any preview information.
2. We generate mission planning policies for a UAV in UDE using DP for ReRFSM. Using DP, we generate policies for every state, allowing robust mission planning in UDE.
3. We generate mission planning solutions for a UAV in UDE using a heuristic, LBFS for ReRFSM. Using LBFS, we generate a solution quickly for the current state for large scale problems.
4. We introduce patrolling policies for multiple UAVs in UDE using DP for ReRFSM: By modifying the inputs of ReRFSMs, we obtain policies for multiple UAVs. We consider the two-UAV case as a proof of concept.

1.4. *Organization*

The organization of this paper is as follows. Section 2 presents the theoretical background of this paper: FSM, FSM composition and pruning, Restricted Finite State Machine (RFSM), and ReRFSM. Section 3 describes how we design UDE and UAV missions in UDE using ReRFSM. Section 4 formulates the problem and Sec. 5 describes how we generate and update the policy or solution, and provides the algorithms. Section 6 analyzes the resulting plan, computation time, and cost for different settings, and Sec. 7 shows the primary simulation results. Finally, conclusions and future work are stated in Sec. 8.

2. Theoretical Approach

This section contains theoretical background that is needed for the discussion in the remainder of the paper, namely, FSM definition, pruning and composition of FSMs are summarized. RFSM is introduced and ReRFSM are defined.

2.1. *Finite state machines*

An FSM is a computational formalism that can be used to describe the behavior of some aspects of a system. Each state represents a period of time in which the system behaves a certain way. Transitions cause the system to move from one state to another. Transitions may be labeled with 'events' or 'inputs' that trigger them. The FSM might also generate 'outputs' or execute 'actions,' either when executing a transition or on entrance or exit from a state. An FSM is a six-tuple:

$$\text{FSM} = (X, U, Y, f, g, x_0), \tag{1}$$

where X, U, and Y are sets, f and g are functions, and $x_0 \in X$. Here, X is the state space, U is the input alphabet, Y is the output alphabet, $f: X \times U \to X$ is the next state function, $g: X \times U \to Y$ is the output function, and $x_0 \in X$ is the initial state.

The interpretation of f and g is as follows: if $x(k) \in X$ is the current state at step k and $u(k) \in U$ is the current input signal, then the current output symbol $y(k)$ and the next state $x(k+1)$ are given by

$$x(k+1) = f(x(k), u(k)), \tag{2}$$

$$y(k) = g(x(k), u(k)), \tag{3}$$

and $x(0)$ is x_0.

To represent an FSM, we can use the sets and functions models as given above, state transition diagrams or state transition tables.

In an FSM, an absorbing state is a state from which there is no transition to another state. Hence, if the state of the FSM reaches an absorbing state, then the state remains there perpetually, regardless of the input.

2.2. *Finite state machine composition and pruning*

We consider the following composition operation on FSMs. Given a number w of FSMs, $\text{FSM}_1 = (X_1, U_1, Y_1, f_1, g_1, x_{10})$, $\text{FSM}_2 = (X_2, U_2, Y_2, f_2, g_2, x_{20}), \ldots, \text{FSM}_w = (X_w, U_w, Y_w, f_w, g_w, x_{w0})$, we define the composition of $\text{FSM}_1, \text{FSM}_2, \ldots, \text{FSM}_w$, denoted $\text{FSM}_1 \times \text{FSM}_2 \times \cdots \times \text{FSM}_w$, as the FSM:

$$\begin{aligned} &\text{FSM}_1 \times \text{FSM}_2 \times \cdots \times \text{FSM}_w \\ &\quad = (X_1 \times X_2 \times \cdots \times X_w,\ U_1 \times U_2 \times \cdots \times U_w, \\ &\qquad Y_1 \times Y_2 \times \cdots \times Y_w,\ (f_1, f_2, \ldots, f_w), \\ &\qquad (g_1, g_2, \ldots, g_w), (x_{10}, x_{20}, \ldots, x_{w0})), \end{aligned} \tag{4}$$

where $\text{FSM}_1, \text{FSM}_2, \ldots, \text{FSM}_w$ are called the components of the composed FSM.

We also consider the following pruning operation on composed FSMs. If one of the component FSMs is not

needed, we remove that FSM as follows. Assume that in (4), FSM_i is not needed. Then, we define the pruning of FSM_i from the FSM, denoted FSM/FSM_i, as the FSM:

$$\text{FSM}/\text{FSM}_i = \text{FSM}_1 \times \cdots \times \text{FSM}_{i-1} \times \text{FSM}_{i+1} \times \cdots \times \text{FSM}_w. \tag{5}$$

2.3. *Restricted finite state machines*

To represent realistic situations, specifically resource or ability limitations, some of the transitions corresponding to the constraints may be restricted during the composition in (4). We call this composed FSM an RFSM. Thus, the RFSM is a composed FSM with the component FSMs defined as (4) but the input alphabet, U:

$$\begin{aligned} & U \subset U_1 \times U_2 \times \cdots \times U_w, \\ \text{where} \quad & U \neq U_1 \times U_2 \times \cdots \times U_w, \end{aligned} \tag{6}$$

and/or

$$\exists x \in X, \quad u \in U : f(x(k), u(k)) = g(x(k), u(k)) = \emptyset, \tag{7}$$

where $\emptyset$ means empty, so the transition is not available. In other words, we disable transitions that correspond to insufficient or excessive resource or capability use. This is an example of supervisory control [21, 22]. The limitations are different for different situations/problems, so restriction rules for the transitions are problem dependent. In general, for the case of the composition of w component FSMs with input restrictions, the RFSM is denoted as

$$\text{RFSM} = \text{FSM}_1 \ \bar{\times} \ \text{FSM}_2 \ \bar{\times} \ \cdots \ \bar{\times} \ \text{FSM}_w, \tag{8}$$

and the pruning of FSM_i from the RFSM is denoted as

$$\text{RFSM} = \text{RFSM} \ \bar{/} \ \text{FSM}_i. \tag{9}$$

2.4. *Recomposable restricted finite state machines*

A ReRFSM is an RFSM that has three modes of operation: normal, composition and pruning.

(a) In the normal mode of operation, the ReRFSM operates as an RFSM.

(b) The composition mode happens when a new component FSM arises. Assume that this happens at step k; let $\text{RFSM}(k)$ be the RFSM in which the ReRFSM operates at step k, and let FSM_{new} be the new component FSM that arises at step k. Then, we have:

$$\text{RFSM}(k+1) = \text{RFSM}(k) \ \bar{\times} \ \text{FSM}_{\text{new}}. \tag{10}$$

(c) The pruning mode happens when a component FSM is not needed. Assume that at step k, component FSM_i is not needed. Then, we have:

$$\text{RFSM}(k+1) = \text{RFSM}(k) \ \bar{/} \ \text{FSM}_i. \tag{11}$$

3. Mission Design

Our design is based on two different state machines: the waypoints FSM, and the task FSM. The waypoints FSM represents the position of the waypoints and starting point as well as paths between them. The task FSM represents tasks to be performed at each waypoint. For a complete system representation, we compose the waypoints FSM and the m task FSMs using the composition operation with input restrictions to get the RFSM. In the following sections, we describe the waypoints FSM, task FSM, and restricted composition for single as well as multiple UAVs.

3.1. *Example problem description*

Consider the following scenario: One UAV is flying in a geographic area on a patrolling mission, and starts off with three waypoints to visit. The waypoints have different priorities. There is a confrontation in the area, resulting in a no-fly zone over which the UAV may not fly (for instance, because of risk, operations by friendly aircraft, or enemy actions). Figure 1 shows the example scenario. In the left subfigure, which shows the initial environment, there are three waypoints to check, each indicated by a star. One path is not available because of dangerous flight conditions. Based on the situation, the UAV generates an initial plan, and it starts to visit the first waypoint according to the initial plan. However, this is the UDE nature of the scenario, when the UAV visits the first waypoint, priorities of the remaining two waypoints are changed, new waypoints are added and availability (or unavailability) of paths may also change with time along some tours, as shown in the right subfigure of Fig. 1. Thus, the UAV has to modify its initial plan to accomplish the mission. Surveillance or reconnaissance missions [23, 24] are instances of this example problem description; they may require that waypoints be added or deleted, and the priorities of the waypoints be changed based on the collected information of the visited waypoints. We aim, in this paper, to develop a mission planner that can deal with the dynamic nature of such a problem.

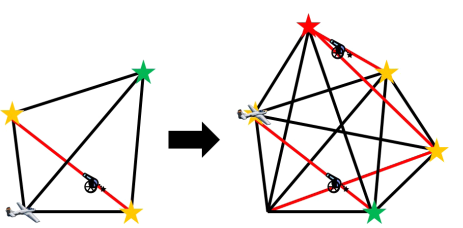


Fig. 1. Example of UAV patrolling mission in an UDE.

3.2. *Waypoints FSM*

In the waypoints FSM, states are the position of a waypoint or starting point, and transitions represent all available paths between the waypoints and starting point. One can think of the waypoints FSM as a directed graph. Given the information on no-fly zones, unavailable paths can be removed from the directed graph by deleting the arcs that cross over the no-fly zones. We assign distance between any two waypoints as the cost of the corresponding transition. An example of waypoints FSM for a single UAV and three waypoints is shown in Fig. 2.

States, or vertices on the graph, represent the waypoints and the starting point. The input alphabet is: S for 'stay' and M_i for 'move to state i.' The initial state $i = 0$ is the starting point. Note that there is no transition between states 1 and 2 because the corresponding path crosses over a no-fly zone. The waypoints FSM for the example shown in Fig. 2 can be represented as a four-tuple:

$$\begin{aligned} X_W &= \{0, 1, 2, 3\}, \\ U_W &= \{S, M_0, M_1, M_2, M_3\}, \\ x_{W_0} &= 0, \\ x_W(k+1) &= f_W(x_W(k), u_W(k)), \end{aligned} \tag{12}$$

where f_W operates as shown in Table 1.

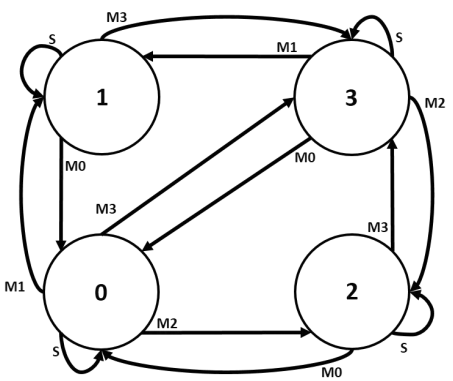


Fig. 2. Waypoints FSM for a case with a single UAV and three waypoints.

Table 1. Waypoints FSM state transition table.

	State 0	State 1	State 2	State 3
Input S	State 0	State 1	State 2	State 3
Input M_0	N/A	State 0	State 0	State 0
Input M_1	State 1	N/A	N/A	State 1
Input M_2	State 2	N/A	N/A	State 2
Input M_3	State 3	State 3	State 3	N/A

3.3. *Task FSM*

At each waypoint, the task FSM represents a task to be performed. For simplicity, we restrict tasks at all waypoints to a simple 'check' task. An example of task FSM for a single waypoint is shown in Fig. 3.

For state and output alphabets, Done represents a checked waypoint and 0 an unchecked waypoint. The input alphabet is: C for 'check waypoint' and NC for 'do not check waypoint'. The initial state is 0 for all waypoints. The task FSM can be represented as a four-tuple:

$$\begin{aligned} X_T &= \{0, \text{Done}\}, \\ U_T &= \{C, NC\}, \\ x_{T_0} &= 0, \\ x_T(k+1) &= f_T(x_T(k), u_T(k)), \end{aligned} \tag{13}$$

where f_T operates as shown in Table 2.

3.4. *Mission RFSM*

In order to form a UAV RFSM representing a complete mission for a single UAV checking m waypoints, we compose a waypoints FSM with m task FSMs (one task FSM for each waypoint) using the composition operator defined earlier with input restrictions. For n UAVs, we compose a UAV FSM n times (one for each UAV).

We choose a particular style of composition called reactive/synchronous FSMs. Our composition is reactive since it reacts to external stimulus. Synchronous style dictates that each state machine in the composition reacts simultaneously and instantaneously. Thus, a reaction of the composite machine consists of a set of simultaneous reactions of each of the component state machines. In the following, we discuss the complete mission model for single and multiple UAVs.

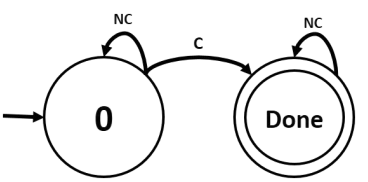


Fig. 3. Task FSM for single waypoint.

Table 2. Task FSM state transition table.

	State 0	State Done
Input NC	State 0	State Done
Input C	State Done	N/A

3.4.1. *Single UAV*

Let us consider our earlier example of three waypoints and a starting point to compose a mission RFSM for single UAV. As explained earlier, we have to compose one waypoint FSM with three task FSMs (in this case $m = 3$) with input restrictions to form a mission RFSM. Inputs for the composed FSM are all possible combinations of inputs for each FSM. However, some restrictions should be applied for the RFSM. Thus, the mission RFSM for a single UAV is as follows:

$$\begin{aligned} X_U &= X_W \times X_{T_1} \times X_{T_2} \times X_{T_3}, \\ U_U &= \{u_U \in U_W \times U_{T_1} \times U_{T_2} \times U_{T_3} : \text{action}(u_U) \leq 1\}, \\ x_{U_0} &= x_{W_0} \times x_{T_{1_0}} \times x_{T_{2_0}} \times x_{T_{3_0}}, \\ f_U &= (f_W, f_{T_1}, f_{T_2}, f_{T_3}), \end{aligned} \tag{14}$$

where

$$\begin{aligned} &f_U(x_U, u_U) = \emptyset, \quad \text{when} \\ &\{u_U = (\bullet, C, \bullet, \bullet) \in U_U, x_U \neq (1, \bullet, \bullet, \bullet) \in X_U\}, \quad \text{and} \\ &\{u_U = (\bullet, \bullet, C, \bullet) \in U_U, x_U \neq (2, \bullet, \bullet, \bullet) \in X_U\}, \quad \text{and} \\ &\{u_U = (\bullet, \bullet, \bullet, C) \in U_U, x_U \neq (3, \bullet, \bullet, \bullet) \in X_U\}, \end{aligned} \tag{15}$$

where $\text{action}(u_U)$ is the number of active actions in the input u_U, the active action is one of $\{M_0, M_1, M_2, M_3, C\}$, and $\bullet$ means all possibilities. The restrictions in (15) mean that the input C for task i is only available at the waypoint i, i.e. the UAV can check the waypoint when the UAV is at the waypoint, and the restrictions in (14) mean that the UAV can only perform one active action at a time.

For example, the input alphabet (S, C, C, NC) is restricted since checking two waypoints concurrently is impossible. Similarly (M_1, NC, NC, C) is prohibited, as moving to waypoint #1 while checking waypoint #3 synchronously is banned. On the other hand, some of the allowed input alphabet elements are $(S, NC, NC, C), (M_2, NC, NC, NC)$, and (S, NC, C, NC) where in the first case the UAV stays at waypoint #3 and starts checking it. In the second input case, the UAV starts heading to waypoint #2. Then the UAV starts checking waypoint #2 as in the third input alphabet case.

3.4.2. *Multiple UAVs*

For n UAVs, the mission RFSM can be represented as:

$$\begin{aligned} X_{MU} &= X_{U_1} \times X_{U_2} \times \cdots \times X_{U_n}, \\ U_{MU} &= \{u_{MU} \in U_{U_1} \times U_{U_2} \times \cdots \times U_{U_n} : \\ &\qquad u_{U_1} \neq u_{U_2} \neq \cdots u_{U_n}\}, \\ x_{MU_0} &= x_{U_{1_0}} \times x_{U_{2_0}} \times \cdots \times x_{U_{n_0}}, \\ f_{MU} &= (f_{U_1}, f_{U_2}, \ldots, f_{U_n}). \end{aligned} \tag{16}$$

Progressing with our example of three waypoints, let us form a mission for two UAVs. For two UAVs, we assume that there are two sets of inputs where each set of inputs is from the single UAV case. Then, the inputs for two UAVs are all possible combinations of the two sets of inputs with a restriction as described in (16): the inputs from the two sets are not allowed to be the same. For instance, $(S, C, NC, NC, S, C, NC, NC)$ is prohibited because the input for the first UAV (first four elements) is the same as the input for the second UAV (last four elements). An example of a possible input alphabet is $(S, C, NC, NC, M_1, NC, NC, NC)$ where the first input element is for the waypoints FSM for the first UAV, the next three input elements are for each task FSM for the first UAV, the fifth input element is for the waypoints FSM for the second UAV, and the other three input elements are for each task FSM for the second UAV. By doing this, each UAV can have its individual input elements on the same set of the waypoints, so the state can be updated by the actions of two UAVs. Three and more UAVs can be treated in a similar fashion, with the caveat that the cardinality of the space and cost of the computations increase accordingly.

4. Problem Formulation

A number n of UAVs are patrolling $\mathcal{A}$, a compact subset of $\mathbb{R}^2$, which represents our UDE. We discretize the environment's geography as a rectangular grid $[0, X_{\text{area}}]\text{X}[0, Y_{\text{area}}]$ to provide a computationally efficient abstraction for planar rigid body motion. The environment has a number of no-fly zones (L), that are described by $L_j = (x_j, y_j, r_j)$ where (x_j, y_j) is center of the no-fly zone j in Cartesian coordinates, and r_j is the radius of no-fly zone j, where $1 \leq j \leq l$. UAVs are required to check m waypoints (w). Each waypoint i is described by $w_i = (x_i, y_i, p_i)$ where (x_i, y_i) is waypoint position in Cartesian coordinates, p_i is the waypoint's priority level, and $1 \leq i \leq m$. The setup is modeled as an ReRFSM using one waypoints FSM and m task FSMs for each UAV as described earlier.

Let $c(x_W, u_W)$ be a function called the cost function, which assigns positive cost (representing travel distance between two waypoints) to the corresponding transition in the waypoints FSM. In addition, each transition in the task FSM has a patrolling performance index as:

$$\begin{aligned} \mathcal{PP}(x_{T_i}, u_{T_i}) &= R, \quad \text{when } x_T = \text{Done}, \\ \mathcal{PP}(x_{T_i}, u_{T_i}) &= p_i \times P, \quad \text{when } x_T = 0, \end{aligned} \tag{17}$$

where R is the reward for a waypoint being in the Done state and P is a penalty for a waypoint being in the 0 state.

In normal mode, the ReRFSM operates as a RFSM with associated transition weights. The RFSM transition weight is the sum of the transition weights of its components. Transition weight in the waypoints FSM is defined as cost

$c(x_W, u_W)$, while transition weight in the task FSM is its patrolling performance $\mathcal{PP}(x_{T_i}, u_{T_i})$. So transition weight in the ReRFSM, $\mathcal{W}(x_U, u_U)$ can be represented as:

$$\mathcal{W}(x_U, u_U) = c(x_W, u_W) + \sum_{i=1}^{m} \mathcal{PP}(x_{T_i}, u_{T_i}). \tag{18}$$

For n UAVs, the transition weight in the ReRFSM, $\mathcal{W}(x_{MU}, u_{MU})$ can be represented as:

$$\mathcal{W}(x_{MU}, u_{MU}) = \mathcal{W}(x_{U_1}, u_{U_1}) + \cdots + \mathcal{W}(x_{U_n}, u_{U_n}). \tag{19}$$

For accessible analysis, we assign negative values to task FSM transition rewards, R in (17). Thus we can define the UDE patrolling problem as minimizing ReRFSM transition weights.

Definition 4.1 (UDE Patrolling Problem). Given that L and w are changing dynamically in an unpredictable manner, find tours such that the UAVs check all waypoints and return to their starting point that minimize the sum of sequences of the transition weights, $\mathcal{W}$.

5. Policy and Solution Approach

5.1. *DP for ReRFSM*

Note that the optimal policy of a composed FSM is obtained by the union of the optimal policies of its component FSMs. However, for RFSMs, this does not hold due to the restricted inputs in RFSM. Thus, the optimal policy of RFSMs has to be obtained by applying DP directly to the whole RFSM. Note also that the global optimal policy of ReRFSM is not obtainable using any methods under our assumption that the future information is not available.

Defining FSM in the state space as:

$$\text{RFSM} = (\mathcal{ST}, \mathcal{IP}, f, \mathit{Init}), \tag{20}$$

where $\mathcal{ST}$ and $\mathcal{IP}$ are sets representing X_{MU} and U_{MU} in (16), respectively, let the function f: $\mathcal{ST} \times \mathcal{IP} \to \mathcal{ST}$ be the next state function, and $\mathit{Init} \in \mathcal{ST}$ be the initial state.

The interpretation of f is as follows: if $st(k) \in \mathcal{ST}$ is the current state at step k and $\mathcal{T}_D(k) \in \mathcal{IP}$ is the current input alphabet, then the next state $st(k+1)$ is given by

$$st(k+1) = f(st(k), \mathcal{T}_D(k)), \tag{21}$$

and $st(0)$ is Init.

Given the state transition mapping (21), consider the objective function

$$J(st(0), \ldots, \mathcal{T}_D(0), \ldots) = \sum_{k=0}^{K-1} \mathcal{W}(st(k), \mathcal{T}_D(k)), \tag{22}$$

where $\mathcal{W}$ is the transition weight and K is the time horizon. A sequence of decisions that optimizes J is obtained through DP based on solving the Hamilton–Jacobi Bellman equation, starting from:

$$J^*(st(k)) = \min_{\mathcal{T}_D(k) \in \mathcal{IP}} \{\mathcal{W}(st(k), \mathcal{T}_D(k)) + J^*(f(st(k), \mathcal{T}_D(k)))\}, \tag{23}$$

where J^* is the optimal cost-to-go from state st. From (21) and (23),

$$J^*(st(k)) = \min_{\mathcal{T}_D(k) \in \mathcal{IP}} \{\mathcal{W}(st(k), \mathcal{T}_D(k)) + J^*(st(k+1))\}. \tag{24}$$

Equation (24) illustrates that DP proceeds backwards with respect to steps. The optimal decision is then

$$\mathcal{T}^*_D(st(k)) = \operatorname*{argmin}_{\mathcal{T}_D(k) \in \mathcal{IP}} \{\mathcal{W}(st(k), \mathcal{T}_D(k)) + J^*(st(k+1))\}. \tag{25}$$

By solving DP for every step k for all states $st \in \mathcal{ST}$, we obtain a sequence of optimal transitions ($\mathcal{T}^*_D$) according to $\mathcal{W}$, and this is the optimal policy (π^*) [25, 26]. Then, the optimal policy for FSM is

$$\pi^* = \text{DP}(\text{RFSM}, \mathcal{W}, K), \tag{26}$$

where DP is the dynamic programming operator.

Note that if there is no penalty for being in a given state at step $K+1$ or if there is no penalty for taking a given decision at step K, the optimal cost-to-go for state st at step $(K+1)$, $J^*(st(K+1))$, is zero for all states $st \in \text{ST}$.

DP provides a policy. We use that policy until composition occurs. Then, the ReRFSM and weights are updated and we recompute the policy. When pruning occurs, we keep the same policy.

ReRFSMs allow the state space of a FSM to change dynamically to follow the environment as defined earlier. When new waypoints are added, priorities of some waypoints are altered, or availability of some paths are changed, new task FSMs or an updated waypoints FSM are created and composition occurs. When composition occurs, the transition weight of the new component is added to the transition weight of the ReRFSM. Similarly, if a waypoint is checked, and/or the task FSM of that waypoint needs to be pruned, then pruning occurs. When pruning occurs, the transition weight of the pruned FSM is subtracted from the transition weight of the ReRFSM. The algorithm of DP for ReRFSM is shown in Table 3.

5.2. *LBFS for ReRFSM*

The motivation of this section is that DP suffers from the curse of dimensionality. Using DP for ReRFSM is computationally

Table 3. Algorithm of DP for ReRFSM.

$\pi^*(k) \leftarrow$ DP for ReRFSM$(w(k), w(k-1), L(k), L(k-1))$
if $w(k) \neq w(k-1)$ **or** $L(k) \neq L(k-1)$ **then**
$m \leftarrow \text{length}(w(k))$
for $i = 0$ **to** m **do**
if $i = 0$ **then**
$\text{FSM}_i \leftarrow$ waypoints FSM
else
$\text{FSM}_i \leftarrow$ task FSM of $w_i(k)$
end if
end for
$\text{RFSM} \leftarrow \text{FSM}_0 \bar{\times} \text{FSM}_1 \bar{\times} \ldots \bar{\times} \text{FSM}_m$
$\mathcal{W} \leftarrow$ transition weight of RFSM
for $i = 1$ **to** m **do**
if FSM_i needs to be pruned **then**
$\text{RFSM} \leftarrow \text{RFSM} \bar{/} \text{FSM}_i$
$\mathcal{W} \leftarrow$ transition weight of RFSM
end if
end for
$\pi^*(k) = \text{DP}(\text{RFSM}, \mathcal{W}, K)$
else
$\pi^*(k) = \pi(k-1)$
end if
return $\pi^*(k)$

expensive, especially if the number of waypoints is large. In such situations, we may not use DP for ReRFSM for the policy; instead we may find the solution for the current state. In addition to that, if we encounter high UDE, that is, the environment changes frequently and unpredictably, the previous policy is not useful. In such a situation, obtaining a policy is useless, and obtaining a solution may be better.

Thus, we use a Breadth-First Search (BFS) heuristic algorithm with fixed depth (*fd*) to get the solution for the current state. The ReRFSM is recomposed at every time step to take into account the environment changes.

However, if the state space and input space for the ReRFSM are very large, BFS with *fd* may not be feasible. In those cases, we set the upper bound for the computational cardinality, CC_{ub}, for each node for BFS as follows:

$$\text{number of next states} \times \text{number of inputs} \leq CC_{ub}. \quad (27)$$

In other words, the computational cardinality at each node for BFS has to be less than the upper bound, CC_{ub}. We call this LBFS. Once we have the number of inputs, the number of next states is obtained based on (27), and the next state is decided based on the cost, highest first for maximizing, lowest first for minimizing. Consequently, LBFS with *fd* ensures a small amount of computation time. The algorithm for LBFS for ReRFSM is shown in Table 4.

Table 4. Algorithm of LBFS for ReRFSM.

$u(k) \leftarrow$ LBFS for ReRFSM$(w(k), L(k),)$
$m \leftarrow \text{length}(w(k))$
for $i = 0$ **to** m **do**
if $i = 0$ **then**
$\text{FSM}_i \leftarrow$ waypoints FSM
else
$\text{FSM}_i \leftarrow$ task FSM of $w_i(k)$
end if
end for
$\text{RFSM} \leftarrow \text{FSM}_0 \bar{\times} \text{FSM}_1 \bar{\times} \ldots \bar{\times} \text{FSM}_m$
$\mathcal{W} \leftarrow$ transition weight of RFSM
for $i = 1$ **to** m **do**
if FSM_i needs to be pruned **then**
$\text{RFSM} \leftarrow \text{RFSM} \bar{/} \text{FSM}_i$
$\mathcal{W} \leftarrow$ transition weight of RFSM
end if
end for
$u(k) = \text{LBFS}(\text{RFSM}, \mathcal{W}, fd, CC_{ub})$
return $u(k)$

6. Analysis of the Approach

6.1. *Plan*

Given the sets of the waypoints and no-fly zones, the resulting plan changes based on the choice of rewards *R* and penalties *P*. Thus, *R* and *P* need to be carefully chosen to obtain the desired plan. Assume that we have seven waypoints with different levels of priority and no-fly zones as follows: $w = \{(2, 4, 2), (4, 2, 2), (5, 5, 1), (9, 10, 3), (5, 7, 1), (8, 1, 2), (2, 9, 1)\}$ and $L = \{(3, 3, 0.5), (6, 1.5, 1)\}$. Then, we generate plans for the given waypoints for different values *R* and *P* using DP for ReRFSM with $K = 20$, and compare the total lengths of the tours of the plans.

As shown in Fig. 4, the total length of the tour is changed for different *P* but not for different *R*, which indicates *P* is the main driver to determine the plan. For low levels of *P*, from 0.1 to 0.4, the plans yield the minimum length of the tour. For medium levels of *P*, from 0.5 to 1.4, the plans yield longer lengths of tour and for high levels of *P*, that is, larger than 1.4, the plans yield the longest lengths of tour. These results indicate that when the level of *P* is high, the plan emphasizes visiting the high priority waypoints over minimizing the total length of the tour; this results in longer than minimum total tour length. Thus, if minimizing the total tour length is more important, a low level of *P* should be used; if checking high priority waypoints is more important, a high level of *P* should be used; if balancing between the two is needed, a medium level of *P* should be used. These low, medium, high levels of *P* can be obtained based on simulations.

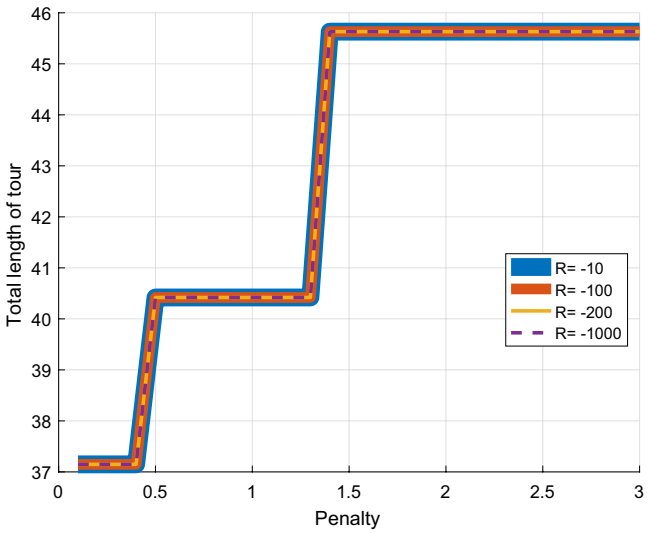


Fig. 4. Total length of tour for different penalties (P) and rewards (R).

6.2. *Computation time and costs*

One of the important questions when using LBFS for ReRFSM is how to determine CC_{ub}. To determine CC_{ub}, computation time, number of waypoints that we can handle, and cumulative cost have to be considered. Thus, the computation time and the cost for the different CC_{ub} for different numbers of waypoints are compared in simulations. Every simulation is performed on the following environment using MATLAB® R2016b: Intel® Core™ i5-4300U Processor CPU 1.90 GHz, 8.0 GB RAM.

For the comparison, we choose $fd = 10$. The cost comparison is shown in the lower figure in Fig. 5. The different colors indicate different CC_{ub} as indicated in the legend, the x axis is the number of waypoints, and the y axis is the cost difference with the case of $CC_{ub} = 3500$ in percentage, so it is zero for $CC_{ub} = 3500$. The cost difference between the case of $CC_{ub} = 3500$ and the others are less then 0.5% for 400 waypoints and less than 4% for up to 1000 waypoints. However, the computation time, as shown in the upper figure in Fig. 5, for the case of $CC_{ub} = 3500$ is less than 10 seconds for 400 waypoints while the computation time for the case of $CC_{ub} = 56{,}000$ is more than 500 seconds. Thus, for the case of fewer than 400 waypoints, which is a large number of waypoints, $CC_{ub} = 3500$ can be used for fast computation time, reasonable cost, and large number of waypoints handling. In this particular choice, the upper bound is around 400 waypoints, but this number can be increased, or decreased, based on the choice of CC_{ub} and desired computation time.

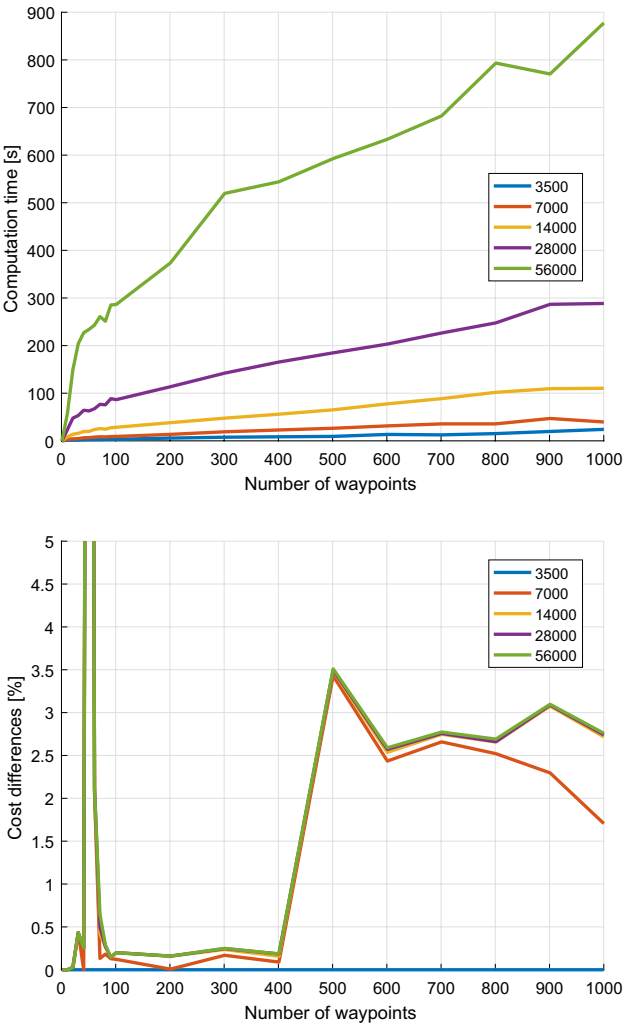


Fig. 5. Comparison of computation times and costs.

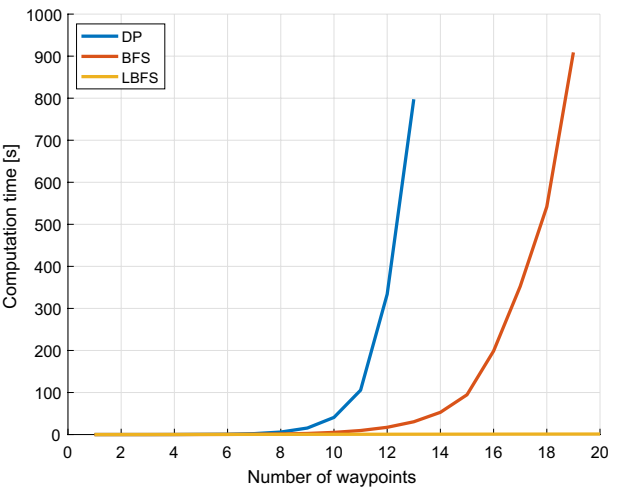


Fig. 6. Comparison of computation times of DP, BFS and LBFS.

Then, the computation times of DP, BFS and LBFS for different numbers of waypoints are compared. For the comparison, we choose $K = 10$ for DP, $fd = 10$ for BFS and LBFS, and $CC_{ub} = 3500$ for LBFS. The computation time comparison is shown in Fig. 6.

As shown in the figure, BFS is slightly faster than DP as DP needs around 100 seconds for 11 waypoints, but BFS needs around 100 seconds for 15 waypoints. However, both computation times increased exponentially, so the methods may not handle large numbers of waypoints in real-time, while the computation time of LBFS is very small compared to DP and BFS, and it can be used in near real-time for large numbers of waypoints.

7. Simulation Results

In this section, every simulation is performed in the environment described in the previous section. For the simulations, we choose $P = 2, R = -100, p_1 = 1$ (low priority), $p_2 = 2$ (medium priority), $p_3 = 3$ (high priority), $K = 10, fd = 10$, and $CC_{ub} = 3500$.

7.1. *Single UAV: Small scale example*

In this section, we follow the example scenario described in the earlier sections, where a single UAV has to check three waypoints that have different priorities. The UDE is considered by adding more waypoints and a no fly zone, and changing the priorities of the waypoints. We then show how the plans are changed based on the no fly zones and priorities of the waypoints. All the plans are obtained by DP for ReRFSM in this section.

At first, the UAV starts from point $(1, 1)$, and has to check three waypoints at $(0.8, 4), (5, 5)$, and $(4, 1)$ that have medium, low, and medium priority, respectively. In addition to that, a no fly zone is located at $(3, 2)$ with radius of 0.5. The UAV has an initial plan as shown in Fig. 7. The length of the tour is 14.45, which is the exact minimum length of the tour for the given waypoints.

Based on the initial plan, the UAV moves to the waypoint $(0.8, 4)$ first to check the waypoint. When the UAV reaches the first waypoint, two new waypoints at $(3, 7)$, and $(6, 3)$ are added, that have high and medium priority, respectively. The priority of the waypoint at $(5, 5)$ is changed from low to medium and that of the waypoint at $(4, 1)$ is changed from medium to low. Another no fly zone at $(4, 6)$ with radius 0.3 is appended. Our method can handle the situation, so a new plan is generated as shown in Fig. 8. Complying with the new no fly zone, priorities, and waypoints added, the total length of the tour is 23.53.

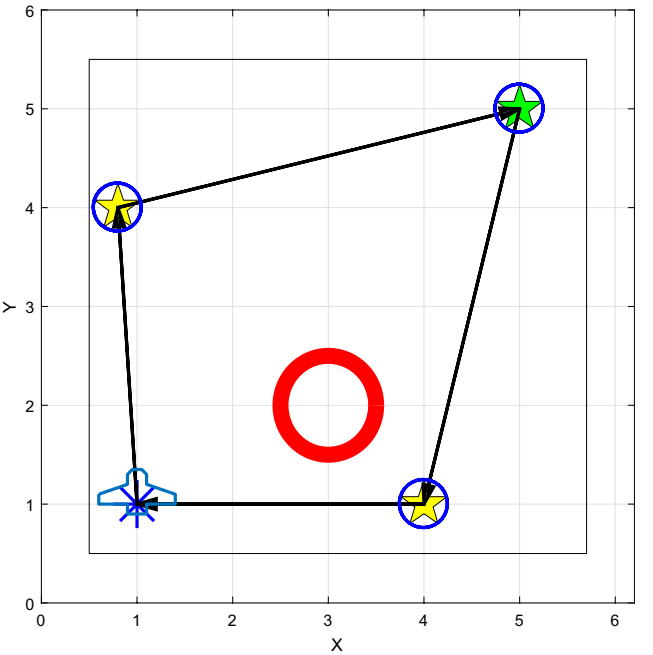


Fig. 7. Initial plan.

7.1.1. *Initial plan with different settings*

The initial plans, without priorities, and without an unavailable path, are shown in Fig. 9. When there is no priorities as shown in the left subfigure of Fig. 9, the tour is similar to the

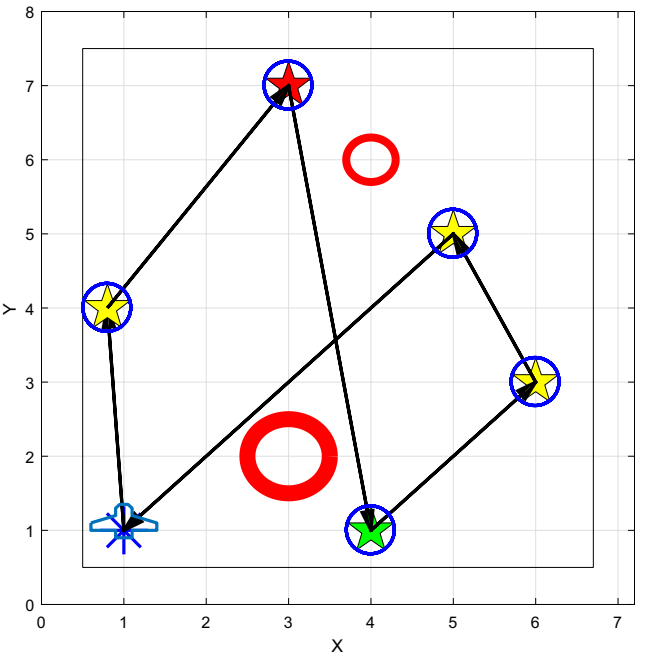


Fig. 8. Updated plan.

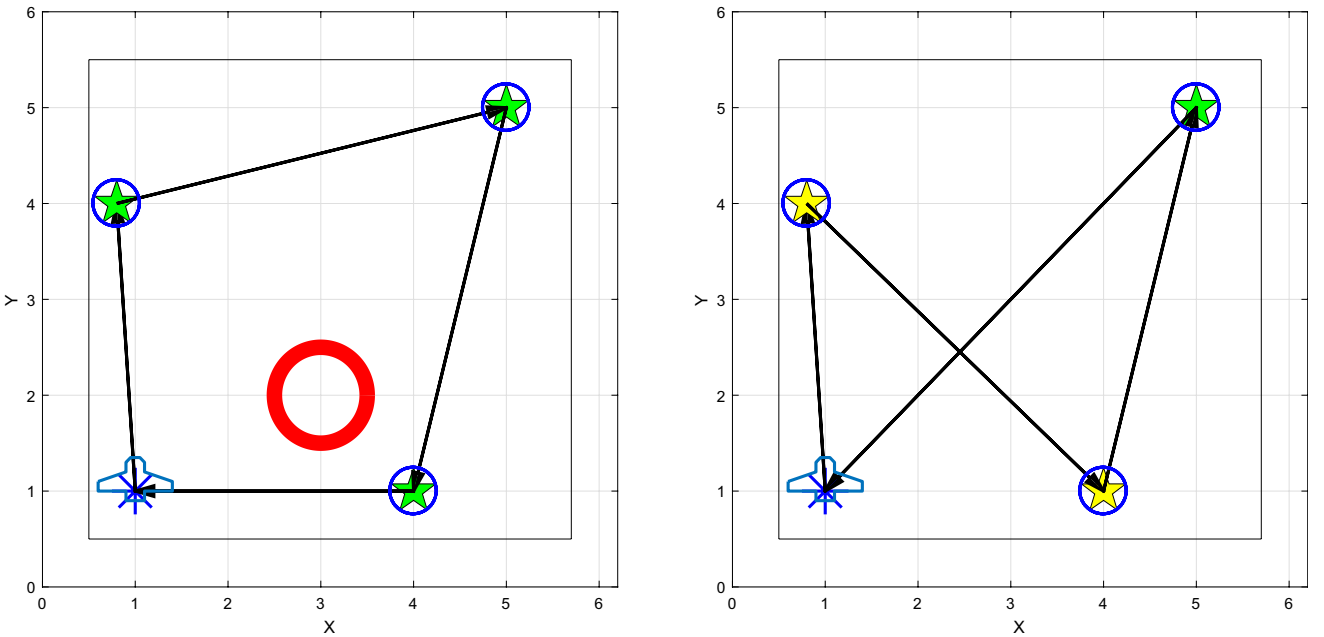


Fig. 9. Initial plans with different settings: without priorities and without no fly zones.

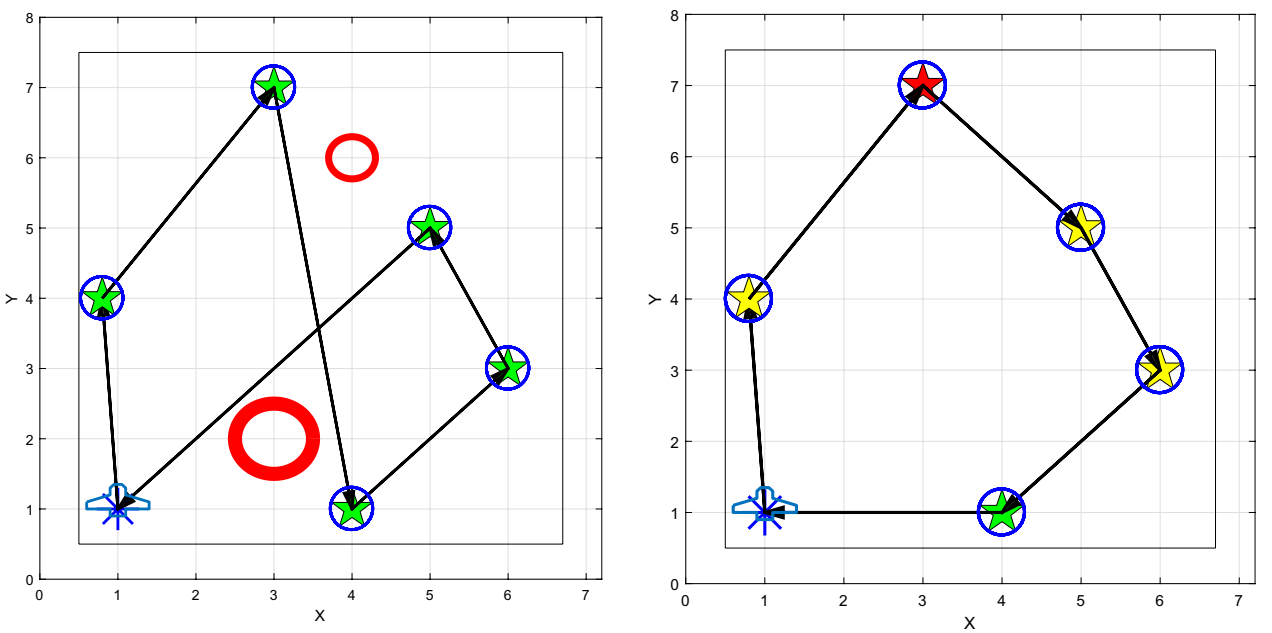


Fig. 10. Updated plans with different settings: without priorities and without no fly zones.

initial plan with priorities and an unavailable path, so the length of the tour is the same. When all paths are available as shown in the right subfigure of Fig. 9, the UAV visits high priority waypoints first, so the total length of the tour, 17.17, is larger than in the other two cases. The length of the tour can be affected by the priorities of the waypoints, so the priorities of the waypoints have to be considered.

7.1.2. *Updated plan with different settings*

The updated plan, without priorities and without unavailable paths, is shown in Fig. 10. When there is no priority as shown in the left subfigure of Fig. 10, the UAV looks for the minimum tour, but because of unavailable paths, the resulting tour is not the exact minimum tour for the given

waypoints, and the resulting tour length is 23.53 which is the same as the updated plan in Fig. 8. When all paths are available, as shown in the right subfigure of Fig. 10, the UAV visits high priority waypoints first while minimizing length of tour. The resulting plan is the exact minimum tour for the given waypoints where the length of the tour is 17.62. As is apparent, the no fly zones affect to the plan, so no fly zones have to be considered for the plan.

7.2. *Single UAV: Small scale comparison*

In this section, we run 50 simulations for a small scale problem to compare the DP for ReRFSM and LBFS for ReRFSM. Each small scale problem initially contains eight waypoints with different priorities where the locations and the priorities of the waypoints are randomly decided. During each simulation, one random waypoint is added at time step five, one or two existing waypoints are deleted at time step nine, and two random waypoints are added at time step 13 without any preview information. We do not consider no fly zones in this section. As a benchmark, we also compare our methods with the global optimal solution assuming that perfect knowledge of the future is available, even though this is not the scenario that the methods developed in this paper. The simulation results are shown in Table 5.

The optimal solution is obtained based on DP with perfect future knowledge, so it only requires one computation, but its computation time is large. The optimal solution never visits the waypoint(s) that will be deleted and it knows when/where new waypoints are added, so the average tour length is about 10% shorter than using our methods. However, note that environmental changes are sometimes based on the information collected during patrolling, for example new intelligence, or adversarial actions in response to the patrol, as mentioned previously, so knowing all the future changes at the beginning is not realistic in real world situations.

Four computations occur for DP, that correspond to the initial computation and three for environment changes, but LBFS generates the solution for every time step. However, the average computation time for LBFS is very fast. The average computation time for DP is acceptable, so DP can be used for small scale problems, but if the number of waypoints is further increased, using DP is not feasible. The average tour length is the same for both methods, and both generate the same plans. This indicates that for small scale problems, $fd = 10$ and $CC_{ub} = 3500$ for LBFS generates the same solution as using DP.

Table 5. Comparison of DP and LBFS.

	Optimal	DP	LBFS
Average tour length	21.5831	23.9898	23.9898
Average computation time	438.88 [s]	3.73 [s]	0.17 [s]
Number of computations	once	four times	every step

7.3. *Single UAV: Large scale example*

In this section, we present simulation results for a large scale example using LBFS for ReRFSM. As we mention in the previous section, using DP is not possible for the large scale problem because it is computationally intractable. We show that the LBFS generates the same plans as DP does in the previous section for small scale problems; hence, it is reasonable to use LBFS for large scale problems on which DP cannot be used. We keep in mind that using LBFS for large-scale problems most likely generates sub-optimal solutions. We consider a scenario with 100 waypoints with different priorities, and no fly zones as shown in Fig. 11. In [27], a problem with three waypoints and four UAVs is considered to be small scale, and one with 10 waypoints and eight UAVs is considered to be medium scale. Many UAV patrolling works consider approximately 10 waypoints [3, 4, 18, 23, 24], and commercial UAV autopilots often have a limit on the number of programmable waypoints. Thus, we consider 100 waypoints to be a large-scale example.

For the large scale simulations, we choose $fd = 40$ and $K = 210$. The LBFS for ReRFSM successfully generates a plan, where the length of the tour is 492.73. The average computation time is 11.07 seconds, which is acceptable.

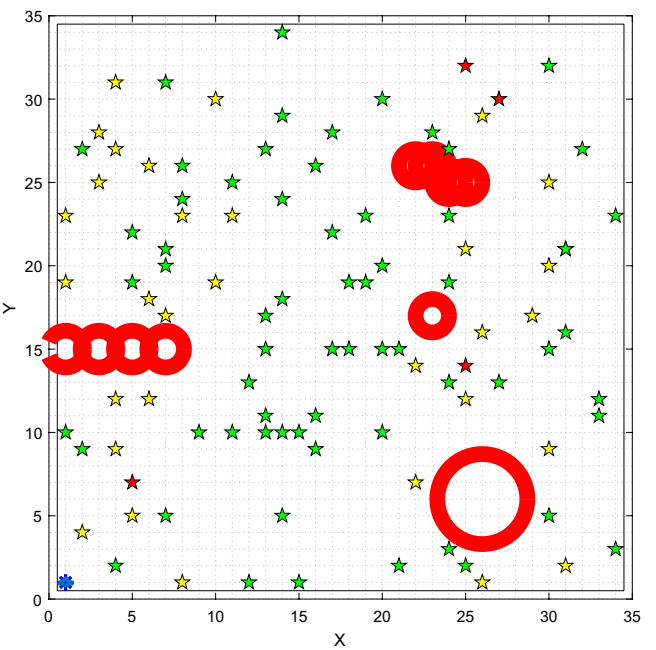


Fig. 11. Large-scale problem example.

Then, we compare the plan with other cases, where the first case does not have priorities for the waypoints, and the second case does not have no fly zones. The length of the tour of the first case is 375.26 and the length of the tour of the second case is 472.52. As expected, the priorities and the no fly zones affect the plan significantly. When there are no waypoint priorities, the UAV looks for the minimum length of the tour, so without priorities, we obtain the shortest tour. Without no fly zones, the UAV wants to visit the high priority waypoints first, so the length of the tour is longer than in the case without priorities, but shorter than in the original case because all the paths are available.

7.4. *Two UAVs: Small scale example*

We simulate a case with two UAVs and four waypoints as shown in Fig. 12 for a proof of concept. Each UAV visits two waypoints and returns to the starting point while avoiding unavailable paths across no fly zones. The first UAV visits the waypoints $(5, 5)$ and $(3.5, 6)$ and then returns to its starting point, which is indicated by a black arrow. The second UAV visits the waypoints $(4, 2)$ and $(2, 4)$ and then returns to $(4, 2)$ and then to the starting point, which is indicated by a blue arrow, because the path between the waypoint $(2, 4)$ and the starting point is not available. The length of the tour for the first UAV is 13.05 and the length of the tour for the second UAV is 11.98.

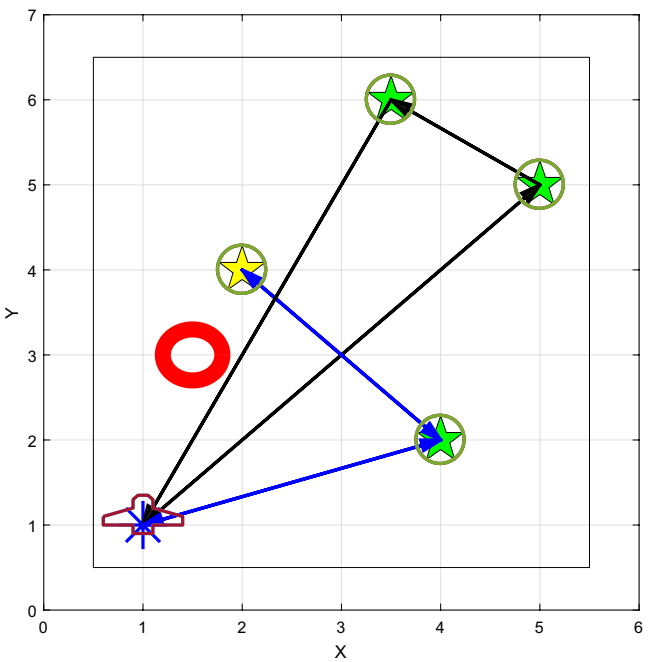


Fig. 12. Plan for two UAVs with four waypoints.

8. Conclusions

A mission planner for a UAV patrolling problem in UDE using DP for ReRFSM and LBFS for ReRFSM is introduced. Our mission planner accounts for an unpredictably changing environment and guides the UAV through it online. The UDE is represented in the FSM framework using ReRFSM. By applying DP for ReRFSM, which can handle approximately 10 waypoints, we generate a robust policy. By applying LBFS for ReRFSM, which can handle large numbers of waypoints, we generate sub-optimal solutions with fast computation times. DP and LBFS for ReRFSM can handle UDE. Our analysis indicates that the gain P has to be carefully chosen to obtain the desired plan: reducing total length of tour or visiting high priority waypoints first. Our simulation results indicate that the priorities and no fly zones are important factors to generate the plan for the waypoints, so they should be considered to obtain the plan for UAV. Finally, we introduce patrolling policies for multiple UAVs in UDE using DP for ReRFSM by modifying the inputs of ReRFSMs. Here, we consider the two-UAV case as a proof of concept.

References

[1] U. S. A. F. C. Scientist, Technology horizons: A vision for air force science and technology 2010-30, tech. rep., United States Air Force (2010).
[2] N. Basilico, N. Gatti and F. Amigoni, Developing a deterministic patrolling strategy for security agents, *IEEE/WIC/ACM Int. Joint Conf. Web Intelligence and Intelligent Agent Technologies*, 2009, pp. 565–572.
[3] K. Barton and D. Kingston, Systematic surveillance for uavs: A feedforward iterative learning control approach, *IEEE American Control Conf.*, June 2013, pp. 5917–5922.
[4] J. L. Fargeas, P. Kabamba and A. Girard, Cooperative surveillance and pursuit using unmanned aerial vehicles and unattended ground sensors, *Sensors* **15** (2015) 1365–1388.
[5] M. Faied, Multistep classification problem using evsi bayesian preposterior framework, *Robot. Auton. Syst.* **72** (2015) 277–284.
[6] S. Lin and B. Kernighan, An effective heuristic algorithm for the traveling salesman problem, *Oper. Res.* **21** (1973) 498–516.
[7] P. Oberlin, S. Rathinam and S. Darbha, Today's traveling salesman problem, *IEEE Robot. Autom. Mag.* **17** (2010) 70–77.
[8] K. Savla, E. Frazzoli and F. Bullo, On the point-to-point and travelling salesperson problem for dubins' vehicle, *American Control Conference*, 2, June 2005, pp. 786–791.
[9] G. Dantzig and J. Ramser, The truck dispatching problem, *Management Sci.* **6** (1959) 80–91.
[10] N. Christofides, The vehicle routing problem, *Revue Francaise D'automatique, D'informatique et de Recherche operationnelle* **10** (1976) 55–70.
[11] G. Laporte, The vehicle routing problem: An overview of exact and approximate algorithm, *Eur. J. Oper. Res.* **59** (1992) 345–358.
[12] M. Figliozzi, Vehicle routing problem for emissions minimization, *Transportation Research Record: Journal of the Transportation Research Board* **2** (2010) 1–7.
[13] H. Psaraftis, Dynamic vehicle routing problem, *Vehicle Routing: Methods and Studies* (1988).

[14] D. Bertsimas and G. V. Ryzin, The dynamic traveling repairman problem, *Sloan School of Management, Massachusetts Institute of Technology* (1989).
[15] K. Savla, F. Bullo and E. Frazzoli, On travelling salesperson problem for dubins' vehicle: Stochastic and dynamic environment, *Conf. Decision and Control, and the European Control Conf.*, December 2005, pp. 4530–4535.
[16] J. Enright, K. Savla and E. Frazzoli, Stochastic and dynamic routing problems for multiple uninhabited aerial vehicles, *J. Guid. Control Dyn.* **32** (2009) 1152–1166.
[17] F. Bullo, E. Frazzoli, M. Pavone, K. Savla and S. Smith, Dynamic vehicle routing for robotic systems, *Proc. IEEE* **99** (2011) 1482–1504.
[18] M. Faied, A. Mostafa and A. Girard, Vehicle routing problem instances: Application to multi-uav mission planning, *AIAA Guidance, Navigation, and Control Conf.*, August 2010, pp. 565–572.
[19] M. Faied and A. Girard, Modeling and optimization of military air operations, in *Proc. 48th IEEE Conf. Decision and Control*, December 2009, pp. 6274–6279.
[20] M. Faied, A. Mostafa and A. Girard, Dynamic optimal control of multiple depot vehicle routing problem with metric temporal logic, *IEEE American Control Conf.* (Jun 2009), pp. 3268–3273.
[21] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd edn. (Springer, 2007).
[22] P. J. Ramadge and W. H. Wonham, Supervisory control of a class of discrete event processes, *SIAM J. Control Optim.* **25** (1987) 206–230.
[23] A. Girard, A. S. Howell and K. Hedrick, Border patrol and surveillance missions using multiple unmanned air vehicles, *IEEE Conf. Decision and Control*, December 2004, pp. 620–625.
[24] A. Matlock, R. Holsapple, C. Schumacher, J. Hansen and A. Girard, Cooperative defensive surveillance using unmanned aerial vehicles, *IEEE American Control Conf.*, June 2009, pp. 2612–2617.
[25] R. R. Weber, *Optimization and Control* (University of Cambridge, 1999).
[26] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, Vol. I, 3rd edn. (Athena Scientific, 2005).
[27] T. Shima and S. Rasmussen, *UAV Cooperative Decision and Control: Challenges and Practical Approaches* (SIAM, 2009).