Resource Management in Constrained Dynamic Situations

by

Jinwoo Seok

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in The University of Michigan
2017

Jinwoo Seok

sjinu@umich.edu

ORCID iD: 0000-0002-4904-726X

To my parents, Yongkee Seok and Youngsun Kim,

To my brother, Jinho Seok,

and

To my wife, Yujin Kim.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincerely appreciation to my advisor, Professor Anouck Girard, for her wisdom, guidance, and support that made it possible to complete my doctoral journey and this dissertation. Without her exceptional advising and understanding, my journey would never have been possible. I also would like to give my sincere thanks to Professor Pierre Kabamba, who passed on a love of researching and teaching. His enthusiasm in research has inspired me, not only in academics, but also in my personal life.

I also would like to thank to Professor Ilya Kolmanovsky for his guidance and feedback on the research. I am privileged to work with him. Thanks to the distinguished faculty members who served on my committee, Professor Kira Barton and Professor Daniel Inman. I really appreciate their valuable time and suggestions. I also thank Dr. Brandon Hencey from the Air Force Research Laboratory for technical discussions and comments.

I acknowledge funding and support from the Office of Naval Research, the Air Force Research Laboratory, and the Boeing Company during my doctoral studies. Thanks for all the support from the Rackham Graduate School and the Department of Aerospace Engineering. Special thanks to the Vehicle Optimization, Dynamics, Control and Autonomy Laboratory members, colleagues, and all the friends.

Finally, the biggest thanks to my father, Yongkee Seok, my mother, Youngsun Kim, my brother, Jinho Seok, and my beloved wife, Yujin Kim for their love, trust, and support.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

LIST OF ABBREVIATIONS

AEA	All Electric Aircraft
ATSP	Asymmetric Traveling Salesman Problem
BFS	Breadth-First Search
DP	Dynamic Programming
DTRP	Dynamic Traveling Repairman Problem
DVRP	Dynamic Vehicle Routing Problem
FAR	Fuel to Air Ratio
FSM	Finite State Machine
HPC	High Pressure Compressor
HPS	High Pressure Shaft
HPSG	High Pressure Shaft Generator
LBFS	Limited Breadth-First Search
LPC	Low Pressure Compressor
LPS	Low Pressure Shaft
LPSG	Low Pressure Shaft Generator
LQR	Linear Quadratic Regulators
MEA	More Electric Aircraft
MMPC	Multiple Model Predictive Control
MPC	Model Predictive Control
MPT	Multi-Parametric Toolbox

NMPC	Nonlinear Model Predictive Control
NPSS	Numerical Propulsion System Simulation
ONR	Office of Naval Research
PI	Proportional-Integral
QoS	Quality of Service
ReFSM	Recomposable Finite State Machine
ReRFSM	Recomposable Restricted Finite State Machine
RFSM	Restricted Finite State Machine
SoC	State of Charge
SWC	Sliding Window Control
T-MATS	Toolbox for the Modeling and Analysis of Thermodynamic Systems
TSP	Traveling Salesman Problem
UAV	Unmanned Aerial Vehicle
UDE	Unpredictably Dynamic Environment
UGS	Unattended Ground Sensor
VRP	Vehicle Routing Problem

ABSTRACT

Resource Management in Constrained Dynamic Situations

by

Jinwoo Seok

Chair: Anouck Girard

Resource management is considered in this dissertation for systems with limited resources, possibly combined with other system constraints, in unpredictably dynamic environments. Resources may represent fuel, power, capabilities, energy, and so on. Resource management is important for many practical systems; usually, resources are limited, and their use must be optimized. Furthermore, systems are often constrained, and constraints must be satisfied for safe operation. Simplistic resource management can result in poor use of resources and failure of the system. Furthermore, many real-world situations involve dynamic environments. Many traditional problems are formulated based on the assumptions of given probabilities or perfect knowledge of future events. However, in many cases, the future is completely unknown, and information on or probabilities about future events are not available. In other words, we operate in unpredictably dynamic situations. Thus, a method is needed to handle dynamic situations without knowledge of the future, but few formal methods have been developed to address them. Thus, the goal is to design resource management methods for constrained systems, with limited resources, in unpredictably dynamic environments.

To this end, resource management is organized hierarchically into two levels: 1) planning, and 2) control. In the planning level, the set of tasks to be performed is scheduled based on limited resources to maximize resource usage in unpredictably dynamic environments. In the control level, the system controller is designed to follow the schedule by considering all the system constraints for safe and efficient operation.

Consequently, this dissertation is mainly divided into two parts: 1) planning level design, based on finite state machines, and 2) control level methods, based on model predictive control. We define a recomposable restricted finite state machine to handle limited resource situations and unpredictably dynamic environments for the planning level. To obtain a policy, dynamic programing is applied, and to obtain a solution, limited breadth-first search is applied to the recomposable restricted finite state machine. A multi-function phased array radar resource management problem and an unmanned aerial vehicle patrolling problem are treated using recomposable restricted finite state machines. Then, we use model predictive control for the control level, because it allows constraint handling and setpoint tracking for the schedule. An aircraft power system management problem is treated that aims to develop an integrated control system for an aircraft gas turbine engine and electrical power system using rate-based model predictive control.

Our results indicate that at the planning level, limited breadth-first search for recomposable restricted finite state machines generates good scheduling solutions in limited resource situations and unpredictably dynamic environments. The importance of cooperation in the planning level is also verified. At the control level, a rate-based model predictive controller allows good schedule tracking and safe operations. The importance of considering the system constraints and interactions between the subsystems is indicated. For the best resource management in constrained dynamic situations, the planning level and the control level need to be considered together.

CHAPTER I

Introduction

1.1 Motivation

Resource management is important in many real-world systems, because, often, resources are limited. Hence, resources have to be utilized optimally. Furthermore, real-world systems are often subject to constraints, for example actuator limits or safety constraints. Resources may include fuel, power, capabilities, or energy, among others. For instance, for a multi-function phased array radar, the available power at each time instance is a resource; for an Unmanned Aerial Vehicle (UAV), the remaining battery charge or the onboard sensors can be viewed as resources. For an aircraft, available fuel is a resource. In addition, the torque of a gas turbine engine can be viewed a resource, and can be thought of as being converted from fuel; the electrical power of a generator can be a resource, and can be thought of as being converted from the torque; finally, the conversion abilities can also be considered as resources. Thus, resource management may include conversions between the resources, and may also require considering efficiencies. In many resource management situations, satisfying all the demands may not be possible because of the limited amounts of available resources, as well as other constraints of the system, such as safety constraints, physical limitations, or other prohibitions. For instance, for a multi-function phased-array radar, the radar is prohibited from transmitting more power than the maximum amount of power for a single task. For an aircraft, a gas turbine engine has to satisfy the surge margin constraints for safe operations, and the conversion ability between the torque and the electrical power is limited by the power of the gas turbine engine and the capacity of the generator. Misguided use of resources can result in poor resource use as well as system failure. Consequently, resource management should consider constraints to maximize the performance and safety of systems. More specifically, by resource management, we mean achieving more and performing better while using limited resources in a constrained system.

Furthermore, many real-world situations involve dynamic environments. By dynamic, we mean that situations change in time in response to decisions – our own, as well as those of others. Many traditional problems are formulated based on the assumptions of given probabilities or perfect knowledge of future events. For instance, Markov process models are based on event probabilities. However, in many cases, the future is completely unknown, and this means that information on or probabilities of future events are not available. In other words, we operate in an Unpredictably Dynamic Environment (UDE). By unpredictable, we mean that we cannot perfectly estimate the changing situation. In many real-world situations, knowing or estimating the future may be challenging. For instance, the systems may be too complicated, or too interconnected to predict, such as in biology, earthquake prediction, or stock market prediction. In an adversarial situation, predicting future behaviors of the enemy is difficult [41, 95] because the behaviors usually depend on humans [15, 63], that are hard to predict at the best of times, and the unobservable space may be large due to the enemy's actions, including denial of sensing in contested areas, obsolete intelligence, and obfuscation. Furthermore, the enemy may try to actively deceive the friendly force, so the predictions (for example, of probabilities) may be poor or completely wrong. Helmuth von Moltke the Elder said "No battle plan survives contact with the enemy." Decisions based on poor or wrong predictions can cause irreparable results in adversarial situations. Thus, we consider dynamic environments, and let situations change in time in response to decisions – our own, as well as those of others, without any knowledge of the future. Methods are needed to handle dynamic situations without predictions of the future, but few formal methods have been developed to address these situations.

As mentioned, the resources are limited and the environments are changed unpredictably in many real-world situations and engineering problems. For instance, a radar or an UAV may have a set of tasks to perform in an UDE; these tasks have to be planned carefully considering the resource limitations and the environments. Resource management for planning belongs to the broad class of task allocation (who does what?) and scheduling (when?) problems. For an aircraft, the scheduled resources, thrust and electrical power, have to be supplied, so uses and conversions of the resources have to be considered to operate the system. Resource management for control is analogous to an operating system, that supplies all the scheduled resources and satisfies a variety of constraints for safe operation of the system. Thus, resource management can be categorized into two broad levels: 1) planning and 2) control. The planning level schedules a set of tasks, in a constrained UDE, to maximize performance subject to limited resources. The control level problem is to design a system controller that is able to follow the task schedule while considering all the system constraints for safe and efficient operations. The planning level and control level need to be considered together for the best resource management in constrained dynamic situations.

Consequently, this dissertation contains two main parts: 1) methods for resource management at the planning level, based on Finite State Machine (FSM) and 2) methods for resource management at the control level, based on Model Predictive Control (MPC). A method is needed to model unpredictably dynamically changing environments easily and precisely. Thus, for the planning level resource management, Recomposable Restricted Finite State Machine (ReRFSM) is defined based on FSM; it can handle limited resource situations by prohibiting some of the transitions in an FSM, and UDE by allowing the state space of an FSM to change dynamically to follow the environment. In an UDE, obtaining local optimal policies of the ReRFSM does not guarantee global optimality, so heuristic methods can be applied to the ReRFSM to obtain solutions.

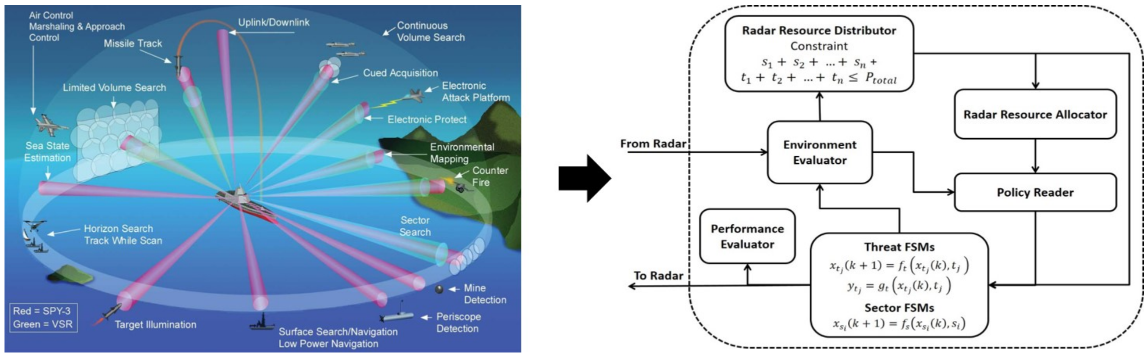

Figure 1.1: Overview of multi-function phased array radar task scheduling.

As shown in Fig. 1.1, in our first case study, the system under consideration is a multi-function phased array radar on a ship. The ship is located in a two dimensional geographic area, surrounded by many threats. The threats are moving toward the radar (or ship) with constant speeds. The limited resource is power emitted from the radar; we treat the resource management as a task scheduling (planning level) problem. This adversarial situation is in an UDE because the set of the tasks to be performed changes in time based on the enemy's behaviors, that are hard to predict; hence, the set of tasks to schedule changes in an unpredictable manner. The goals are to ensure zero leakage, where the radar should be aware of the threats if they are within a certain range, and to discriminate as many threats as possible. This problem was motivated by the Office of Naval Research (ONR) and their interest in studying cooperative, fleet level resource management of multi-function phased array radars onboard ships. Our approach allows for the convenient design of a distributed, cooperative, multi-radar resource management system. Simulation results indicate that heuristic methods generate effective radar schedule solutions in an overwhelming situation.

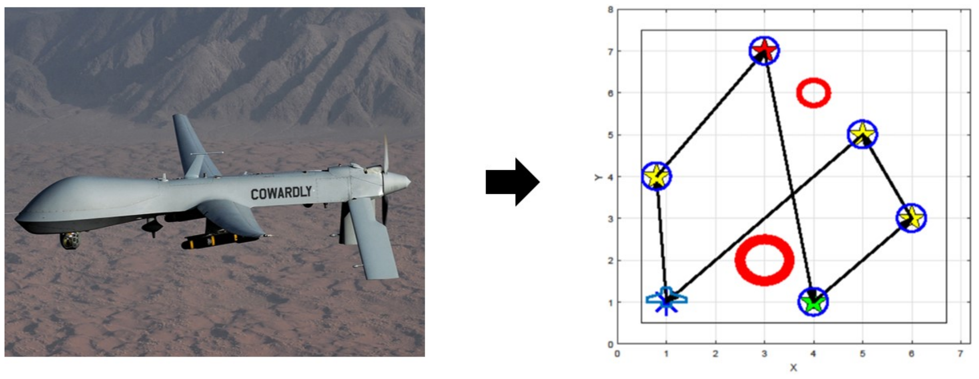

Figure 1.2: Overview of an UAV patrolling mission planner.

As a second case study, the system under consideration is an UAV. The UAV flies over a contested area to gather information on an area of interest (waypoint), as shown in Fig. 1.2. The limited resource is the ability of the UAV, that is, the UAV only can perform one task at a specific location at a time. For instance, the UAV can only gather information about a waypoint when it is physically at the waypoint. Constraints include availabilities of the paths between the areas of interest. We treat the resource management as a task scheduling (planning level) problem. The patrolling mission is in an UDE because the areas of interest may change in time without predictions. For instance, an UAV may collect new, previously unknown information during a patrolling mission, and the information may indicate that the UAV no longer has to visit areas it was previously assigned to, but may have to visit new areas of interest instead. The priorities and/or the risk in some areas of interest may evolve in time in the same unpredictable manner; the set of tasks changes in time in an unpredictable manner. The goal is to find a tour such that the UAV visits all waypoints while minimizing the total travel distance and maximizing surveillance performance based on the priorities of the waypoints. This problem was motivated by many real-world

UAV patrolling missions, such as surveillance for crime prevention, searching for objects or facilities, reconnaissance of contested areas, and so on. The effects of system gains on the resulting plan are analyzed and a two-UAV example problem is treated to study the cooperation of UAVs. Simulations of different settings verify that the mission planner based on heuristic methods successfully generates patrolling plans in UDEs and handles large numbers of waypoints with fast computation time.

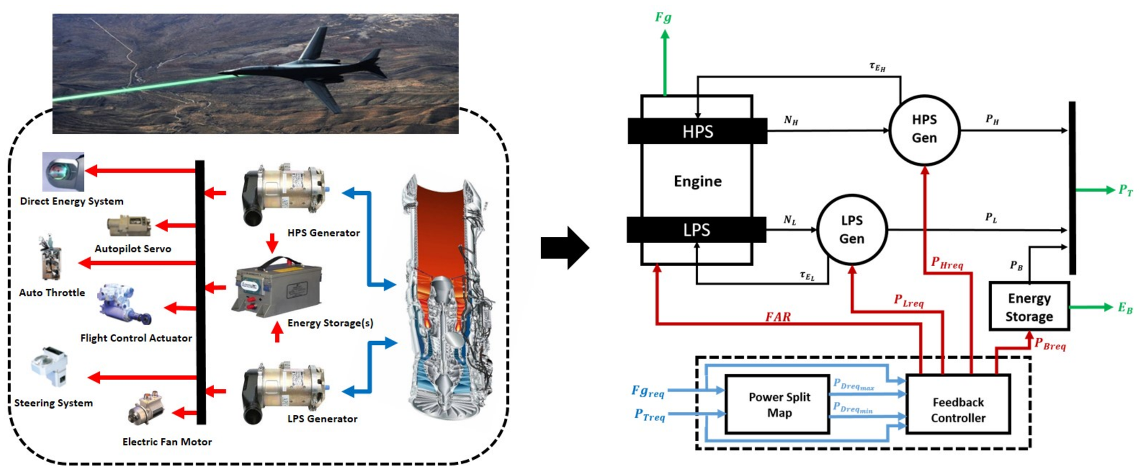

Figure 1.3: Overview of aircraft power system management.

Finally, the third case study considers an aircraft power system that includes a gas turbine engine and electrical power systems as shown in Fig. 1.3. The system is required to supply scheduled thrust and large electrical loads. The limited resources are fuel, electrical power of generators and energy storage elements, as well as the conversion abilities between the resources. Constraints include the gas turbine engine constraints (e.g., surge margins) and electrical system constraints (e.g., component power limits). Thus, we treat the resource management as a system operation (control level) problem. The problem is in an UDE because the schedule can change unexpectedly, so the controller must safely operate the system and follow the schedule in the presence of transient requests without any preview. The goal is to design an integrated rate-based MPC controller that accommodates large steady and transient electrical loads, maintains aircraft flight performance by delivering scheduled thrust, enforces gas turbine engine constraints, as well as electrical system constraints, and reduces fuel consumption. This problem was motivated by future More Electric Aircraft (MEA) or All Electric Aircraft (AEA). An MPC approach is used because the MPC allows constraints handling and set-point tracking (schedule tracking). The system constraints and the interactions between the subsystems have to be considered to follow the schedule safely and efficiently. To alleviate the interaction effects, an advanced two-generator configuration and high performance energy storage elements are considered. For the two-generator configuration, a power split map between the two generators is developed and for the high performance energy storage elements, supervisory logic is developed. Auxiliary offset states are introduced to reduce the mismatch between the linear prediction models and the actual nonlinear system. Simulations results are included to compare the different settings and show that the integrated rate-based MPC controller allows good tracking of thrust and electrical power schedules, and satisfies a variety of constraints for safe operations.

1.2 Contributions

The original contributions of the dissertation are as follows:

- Resource management of constrained systems, with limited resources, in UDEs, is treated in this dissertation. The notions of resource, resource management in constrained environments, and dynamic environments are defined. Resource management is partitioned into two broad categories; 1) planning and 2) control. The planning level maximizes the usage of the limited resource. The importance of cooperation is shown. At the control level, good tracking of the schedule and safe operations considering a variety of system constraints are required. The importance of taking into account the system constraints and the interactions between subsystems is shown. The importance of considering the planning and control levels together in resource management in constrained dynamic situations is discussed.

- The Recomposable Restricted Finite State Machine, ReRFSM, is defined, and can handle limited resource situations in UDEs for planning level resource management. The optimality of the ReRFSM is not achieved by obtaining the local optimal policies of the ReRFSM, due to the unpredictable nature of the environment. Thus, heuristic methods can be applied to the ReRFSM to obtain a solution for planning level resource management in an UDE.

- A multi-function phased array radar task scheduler, with limited radar resources, in an UDE, is designed for planning level resource management using ReRFSM. Because of the UDE, heuristic methods are applied to generate a radar scheduling solution, and the resulting schedule performs well. A distributed, cooperative multi-radar system, using communications between the radars, is designed and the importance of cooperation at the planning level for resource management in UDEs is verified.

- A patrolling mission planner for an UAV in an UDE is designed for planning level resource management using ReRFSM. Because of the UDE, heuristic methods are applied to generate an UAV patrolling plan, and the resulting plan performs well. The development of a multi-UAV patrolling mission planner is also considered to study cooperation of the UAVs at the planning level of resource management.

- A coordinated rate-based MPC controller for aircraft gas turbine engine and electrical power system is designed for control level resource management. The system constraints and the interactions between the subsystems are considered, and the importance of considering them is verified. The integrated rate-based

MPC controller allows good tracking of schedules, and satisfies a variety of constraints, so the system is safely and efficiently operated.

1.3 Dissertation Outline

The outline of this dissertation is as follows. Chapter II presents the theoretical foundations of ReRFSM. The ReRFSM is defined and the optimality of the ReRFSM is analyzed. Chapter III designs a multi-function phased array radar task scheduler in an UDE for planning level resource management using ReRFSM. The method allows the convenient design of a cooperative, distributed resource management system for multi-radar. Simulation results are presented for the single-radar and multi-radar cases. Chapter IV designs a patrolling mission planner for UAVs in an UDE at the planning level, treating the resource management using ReRFSM. A two-UAV system is introduced and simulation results are presented. Chapter V describes the development of a coordinated rate-based MPC controller for an aircraft gas turbine engine and electrical power system for control level resource management. Simulation comparisons of different settings are presented. Finally, Chapter VI presents our conclusions.

CHAPTER II

Recomposable Restricted Finite State Machine

2.1 Introduction

2.1.1 Motivation

The real-world is dynamic and unpredictable. By dynamic, we mean that situations change in time in response to decisions – our own, as well as those of others. By unpredictable, we mean that we cannot perfectly estimate the changing situations. In other words, we operate in UDE. Many traditional problems were formulated based on the assumptions of given probabilities or perfect knowledge of future events. As an alternative, predetermined priority rules may be used. However, in many cases, the future is completely unknown, which means any information or probabilities of the future are not available. In addition to that, predetermined priority rules may not take into account environment changes in the future. Furthermore, in the real-world, the resources are usually limited, and the limitations must be considered when making the decisions.

Many dynamic problems have been formulated; for instance, the Dynamic Vehicle Routing Problem (DVRP) [78] is a dynamic version of the Vehicle Routing Problem (VRP), the Dynamic Traveling Repairman Problem (DTRP) [12] is a dynamic version of Traveling Salesman Problem (TSP), and the dynamic task scheduling problem. By task scheduling we mean given a set of tasks, find the order of execution of the tasks using the given resources. Thus, the task scheduling problem is finding the order of execution of a set of tasks that maximizes or minimizes a desired performance, for example time duration using the given resources. Then, by dynamic task scheduling problem we mean that in the task scheduling problem, we allow the set of tasks to change with time. Dynamic problems and their solution methods are application relevant. As mentioned, many dynamic problems are formulated based on the assumption of knowledge of future events, so many methods are developed and used to solve the problem based on the knowledge. For instance, Markovian process models are based on event probabilities. However, in the case of UDE, where the future is completely unknown, thees methods cannot be used. Thus, methods that can handle UDE are necessary, but most methods depend on predetermined priority rules, and few formal methods have been developed to address them.

Dynamic Programming (DP) for FSM is a well known method and can be applied to static or predictably dynamic problems. FSM are intuitive and easy to use, and DP guarantees an optimal policy. By policy, we mean the solutions for all the states. By solution, we mean the optimal/suboptimal sequence of decisions at the current state. The drawback of DP is the curse of dimensionality. Thus, for large scale problems, a heuristic method can be applied to FSM, such ae Breadth-First Search (BFS), which generates sub-optimal solutions but is computationally effective.

As mentioned previously, the real world is typically an UDE. For instance, in the task scheduling problem, a existing task can be canceled or a new task can be added without any prediction. However, traditional FSM cannot capture changing environments because the state space of the FSM is finite, so only known or predictable states exist. Thus, the FSM needs to be updated according to the changing environment, but changing FSMs in time (dynamics FSMs) requires careful definitions and it was previously unclear what the optimality of such a system is. In this chapter, we define Restricted Finite State Machine (RFSM) based on FSM to handle resource limitations, and Recomposable Finite State Machine (ReFSM) and ReRFSM, which are based on FSM and can handle UDEs, then study the optimality of the resulting systems.

2.1.2 Literature Review

Methods for dealing with dynamically changing environments vary based on the specific problem under consideration. A convenient approach to deal with dynamic environments is rule-based. When the environment changes, a new plan or schedule is obtained based on pre-defined rules. In [98], the authors propose a least-violating control strategy algorithm with safety rules. Each safety rule has an associated priority; thus, the algorithm finds a strategy to reach the goal while minimizing violated priorities. Robotic car navigation in an urban environment is used as an illustrative example. In [65], the authors introduce a distributed play-based role assignment algorithm for teams of robots for the RoboCup four-legged league. Each pre-decided "play" has a role for each robot, so based on the current situation, the "play" is determined, then each robot is assigned a role by the "play."

In [78], the author suggests two approaches to solve the DVRP: the first involves rerunning the whole procedure, and the second considers local updates. Local updates use the previous solution. An example is to insert new tasks into the previous schedule. In [92], the authors solve the DTRP with two priorities, where each task has one of the two priorities. The authors want to minimize expected delay, that is, the time between a task's arrival and its completion. A lower bound is provided and a Randomized Priority policy is proposed. This work is extended to multiple vehicles and multiple priorities in [93]. The authors also provide a lower bound and propose a Separate Queues policy that performs within a constant factor of the lower bound.

In [23], the scheduling of part-feeding tasks of manufacturing lines for a single mobile robot is proposed. The mobile robot has to prevent stopping the manufacturing lines by feeding enough parts to the lines. The authors find the schedule of part-feeding tasks for the mobile robot that minimizes the total traveling time of the robot using mixed integer programming for optimal solutions and a genetic algorithm based heuristic for near optimal but computationally lighter solutions.

Mobile service robot dynamic task scheduling is discussed in [20] and [101]. The authors introduce an online user to a mobile service robots architecture, and schedule the tasks using mixed integer programming, then apply the schedule to an actual mobile service robot called CoBot. The scheduler rejects or asks the user to loosen the constraints if the task is not feasible. It generates a new schedule whenever a new task is requested by the user.

2.1.3 Original Contributions

The original contributions of this chapter are as follows:

1. We define a composition and pruning operations of FSM as well as additive and subtractive costs of associated transitions, then prove that the optimal policy of a composed FSM is the union of the optimal policies of its component FSMs. Thus, the optimal policy of a composed FSM is computationally easily obtained even if the size of the composed FSM is large.

2. We define RFSM and prove that the optimal policy of a RFSM is not the union of the optimal policies of its component FSMs. Thus, the optimal policy of a RFSM, which is a composed FSM with input restrictions, has to be computed directly from the RFSM, so is computationally expensive to obtain if the size of the RFSM is large.

3. We define ReFSM and ReRFSM and investigate the optimality of ReFSM and ReRFSM, and show that the optimal policies of ReFSM and ReRFSM are

not obtained by obtaining the local optimal policies of the composed FSMs in ReFSM and the RFSMs in ReRFSM, thus, in fact, are not obtainable without perfect knowledge of future. We conclude that the current optimal policy (or solution) may not be needed in UDE.

4. We develop a method, Limited Breadth-First Search (LBFS) for ReRFSM, that can handle UDE and is computationally tractable for large scale problems.

2.2 Background

2.2.1 Finite State Machines

An FSM constructs an output signal one symbol at a time by observing an input signal one symbol at a time [32, 16, 60]. An FSM is a six-tuple:

$$FSM = (X, U, Y, f, g, x_0), \tag{2.1}$$

where X, U, and Y are sets, f and g are functions, and $x_0 \in X$. 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:

$$x(k+1) = f(x(k), u(k)), \tag{2.2}$$

$$y(k) = g(x(k), u(k)), \tag{2.3}$$

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. Note that the output alphabet and the output function can be omitted if the output does not exist or is not necessary.

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.

2.2.2 Finite State Machine Composition and Pruning

We consider the following composition operation on FSMs. Given a number w of FSMs, $FSM_1 = (X_1,\ U_1,\ Y_1,\ f_1,\ g_1,\ x_{01})$, $FSM_2 = (X_2,\ U_2,\ Y_2,\ f_2,\ g_2,\ x_{02})$, ..., $FSM_w = (X_w,\ U_w,\ Y_w,\ f_w,\ g_w,\ x_{0w})$, we define the composition of FSM_1, FSM_2, ..., FSM_w, denoted $FSM_1 \times FSM_2 \times\ ...\ \times FSM_w$, as the FSM:

$$FSM_1 \times FSM_2 \times\ ...\ \times FSM_w = (X, U, Y, f, g, x_0), \tag{2.4}$$

where FSM_1, FSM_2, ..., FSM_w are called the components of the composed FSM, and

$$\begin{cases} X & = & X_1 \times X_2 \times\ ...\ \times X_w \\ U & = & U_1 \times U_2 \times\ ...\ \times U_w \\ Y & = & Y_1 \times Y_2 \times\ ...\ \times Y_w \\ f & = & (f_1,\ f_2,\ ...,\ f_w) \\ g & = & (g_1,\ g_2,\ ...,\ g_w) \\ x_0 \in X & = & (x_{01},\ x_{02},\ ...,\ x_{0w}) \end{cases} . \tag{2.5}$$

The interpretation of f and g is as follows: if $x(k) = (x_1(k), x_2(k), ..., x_w(k)) \in X$ is the current state at step k and $u(k) = (u_1(k), u_2(k), ..., u_w(k)) \in U$ is the current input signal, then the current output symbol $y(k)$ and the next state $x(k+1)$ are given by:

$$\begin{aligned} x(k+1) &= f(x(k),\ u(k)) \\ &= \big(x_1(k+1),\ x_2(k+1),\ ...,\ x_w(k+1)\big) \\ &= \big(f_1(x_1(k), u_1(k)),\ f_2(x_2(k), u_2(k)), \\ &\quad ...,\ f_w(x_w(k), u_w(k))\big), \end{aligned} \tag{2.6}$$

$$\begin{aligned} y(k) &= g(x(k),\ u(k)) \\ &= \big(y_1(k),\ y_2(k),\ ...,\ y_2(k)\big) \\ &= \big(g_1(x_1(k), u_1(k)),\ g_2(x_2(k), u_2(k)), \\ &\quad ...,\ g_w(x_w(k), u_w(k))\big), \end{aligned} \tag{2.7}$$

and $x(0)$ is x_0.

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 (2.4), FSM_i is not needed. Then, we define the pruning of FSM_i from the FSM, noted $FSM\ /\ FSM_i$, as the FSM:

$$\begin{aligned} &FSM\ /\ FSM_i = \\ &FSM_1 \times\ ...\ \times FSM_{i-1} \times FSM_{i+1} \times\ ...\ \times FSM_w. \end{aligned} \tag{2.8}$$

We compose the FSMs using a composition operation called synchrony. In this operation, each machine in the composition reacts simultaneously and instantaneously. The synchronous operation and its composition properties are well studied in computer science; see [8, 9, 30, 61, 62].

2.2.3 Recomposable Finite State Machines

An ReFSM is a composed FSM that has three modes of operation: normal, composition and pruning.

a. In the normal mode of operation, the ReFSM operates as a composed FSM.

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

$$FSM(k+1) = FSM(k) \times FSM_{new}. \tag{2.9}$$

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:

$$FSM(k+1) = FSM(k) \ / \ FSM_i. \tag{2.10}$$

2.2.4 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 (2.4). The restrictions may be global, which means some inputs are not available for all the states (input alphabet space is reduced), or local, which means an input in the input alphabet may not available in specific states. We call this composed FSM an RFSM. Thus, the RFSM is a composed FSM with the component FSMs defined as (2.4) but the restricted input alphabet (global input restriction), U:

$$\begin{gathered} U \subset U_1 \times U_2 \times \ ... \ \times U_w, \\ \text{where } U \neq U_1 \times U_2 \times \ ... \ \times U_w. \end{gathered} \tag{2.11}$$

and/or the local input restriction:

$$\exists x \in X, u \in U : f(x(k),\ u(k)) = g(x(k),\ u(k)) = \emptyset, \tag{2.12}$$

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 [16], [80]. 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

$$RFSM = FSM_1 \;\bar{\times}\; FSM_2 \;\bar{\times}\; ... \;\bar{\times}\; FSM_w, \tag{2.13}$$

and the pruning of FSM_i from the RFSM is denoted as

$$RFSM = RFSM \;\bar{/}\; FSM_i. \tag{2.14}$$

2.2.5 Recomposable Restricted Finite State Machines

The ReRFSM is defined as the ReFSM, but with a restricted input alphabet during the composition mode as defined in (2.11) and (2.12). Thus, 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 $RFSM(k)$ 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:

$$RFSM(k+1) = RFSM(k) \;\bar{\times}\; FSM_{new}. \tag{2.15}$$

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:

$$RFSM(k+1) = RFSM(k) \; \bar{/} \; FSM_i. \tag{2.16}$$

2.2.6 Additive and Subtractive Costs

Assume that we have two component FSMs that are FSM_1 and FSM_2 with associated transition costs Φ_1 and Φ_2. When composition of FSM_1 and FSM_2 occurs, the costs are added to each other as follows:

$$\begin{aligned} \Phi_1 + \Phi_2 = \; & \Phi_1(x_1, u_1) + \Phi_2(x_2, u_2) = \Phi((x_1, x_2), (u_1, u_2)), \\ & \text{for all } x_1 \in X_1, u_1 \in U_1, x_2 \in X_2, u_2 \in U_2, \end{aligned} \tag{2.17}$$

where Φ_1, Φ_2 and Φ represent the transition costs of FSM_1, FSM_2, and $FSM_1 \times FSM_2$, x_1 is the state of FSM_1, u_1 is the input of FSM_1, x_2 is the state of FSM_2, u_2 is the input of FSM_2, X_1 is the state space of FSM_1, U_1 is the input alphabet of FSM_1, X_2 is the state space of FSM_2 and U_2 is the input alphabet of FSM_2. We call costs that satisfy the above property additive. Now, assume that we also have the composed FSM that is $FSM_1 \times FSM_2$ with associated costs Φ. When pruning, for instance FSM/FSM_1, occurs, Φ_1 is subtracted from Φ as follows:

$$\begin{aligned} \Phi - \Phi_1 = \; & \Phi((x_1, x_2), (u_1, u_2)) - \Phi_1(x_1, u_1) = \Phi_2(x_2, u_2), \\ & \text{for all } x \in X, u \in U, x_1 \in X_1, u_1 \in U_1, \end{aligned} \tag{2.18}$$

where x is the state of $FSM_1 \times FSM_2$, u is the input of $FSM_1 \times FSM_2$, X is the state space of $FSM_1 \times FSM_2$ and U is the input alphabet of $FSM_1 \times FSM_2$. We call costs that satisfy the above property subtractive.

2.2.7 Dynamic Programming for Finite State Machine

Given the state transition mapping (2.2), consider the objective function:

$$J(x(0), u(0), x(1), ..., u(K-1)) = \sum_{k=0}^{K-1} \Phi(x(k), u(k)), \tag{2.19}$$

where Φ is the transition cost 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:

$$J^*(x(k)) = \min_{u(k)\in U} \{\Phi(x(k), u(k)) + J^*(f(x(k), u(k)))\}, \tag{2.20}$$

where J^* is the optimal cost to-go from state x. From (2.2) and (2.20),

$$J^*(x(k)) = \min_{u(k)\in U} \{\Phi(x(k), u(k)) + J^*(x(k+1))\}. \tag{2.21}$$

Equation (2.21) illustrates that DP proceeds backwards with respect to steps, i.e., the optimal cost and decision at step k depend on the optimal cost and decision at step $(k+1)$. The optimal decision is then

$$u^*(x(k)) = \underset{u(k)\in U}{\operatorname{argmin}} \{\Phi(x(k), u(k)) + J^*(x(k+1))\}. \tag{2.22}$$

By solving DP for every step k for all states $x \in X$, where X is state space, we obtain a sequence of optimal decisions (u^*) according to Φ, and this is the optimal policy (π^*) [104], [10], and [11]. Then, the optimal policy for an FSM is:

$$\pi^* = DP(FSM, \Phi, K), \tag{2.23}$$

where FSM is a finite state machine, Φ is a transition cost, and DP is the dynamic programming operator.

Note that if there is no penalty to being in a given state at step $K+1$, if there is no penalty for taking a given decision at step K, or if K is large enough, the optimal cost to go for state x at step $K+1$, $J^*(x(K+1))$, is zero for all states $x \in X$.

2.3 Problem Formulation

Assume that there are w FSMs with associated transition cost functions and that the FSMs are composed without input restrictions as defined in Sec. 2.2.2 or with input restrictions as defined in Sec. 2.2.4. Assume that the transition costs are additive and subtractive as defined in Sec. 2.2.6. Assume that the number w changes in time in an unpredictable manner. Given the situation, find the best sequence of decisions such that the cumulative cost is minimized.

2.4 Optimal Policy of Composed Finite State Machine

The optimal policy of an FSM is obtained by DP for FSM as described in the previous section. Thus, the optimal policy of a composed FSM is also obtained by DP for the composed FSM. However, if the number of the component FSM for the composed FSM is large, DP may not be computationally feasible to obtain the optimal policy due to the large state space and input space sizes. Thus, in this section, we prove that the optimal policy of a composed FSM is the union of the optimal policies of its component FSMs. We also prove that the optimal policy of a composed FSM after pruning is the separation of the optimal policy of the pruned FSM from the optimal policy of the previous composed FSM. The resulting policy is called as union policy. Thus, the optimal policy of a composed FSM is computationally easily obtained by computing the union policy even if the number of the component FSM is large. Considering a composed FSM that consists of w copies of the same component FSMs. Then, the computation cardinality of DP applied directly to the composed FSM is $(n^w) \times (m^w) \times K$ where n is the number of states of the component FSMs and m is the number of inputs of the component FSMs, while the computation cardinality of obtaining the union policy is $n \times m \times K \times w$, which means obtaining the union policy is $\frac{(n^{(w-1)}) \times (m^{(w-1)})}{w}$ times faster than applying DP directly to the composed FSM.

Without loss of generality, we consider the case of two component FSMs yielding one composed FSM. Then, the union of optimal policies of two FSMs is the optimal policy of the composed FSM and the optimal policy of a composed FSM can be decoupled to optimal policies of two component FSMs.

Consider a ReFSM as defined in Section 2.2.3. Assume that the transitions in each component FSM of the ReFSM have associated cost (typically, assigned heuristically) and that the costs are additive (2.17) upon composition (2.4), (2.9) and subtractive (2.18) upon pruning (2.8), (2.10). The optimal policies are obtained by (2.23) when the composition or the pruning occurs. To prove this, we consider two cases: composition, where the number of states increases, and pruning, where the number of states decreases. We consider each case in turn.

Claim A.

Assume that FSM_1 has cost Φ_1 and FSM_2 has cost Φ_2.

Let $u_1^* = DP(FSM_1, \Phi_1)$, $u_2^* = DP(FSM_2, \Phi_2)$ and $u^* = DP(FSM_1 \times FSM_2, \Phi_1 + \Phi_2)$.

Then, we claim that $u^* = \begin{bmatrix} u_1^* \\ u_2^* \end{bmatrix}$.

Proof A. The optimal cost-to-go of DP for FSM_1 is J_1^*, and it is obtained by (2.20) as follows:

$$J_1^*(x_1(k)) = \min_{u_1(k) \in U_1} \{\Phi_1(x_1(k), u_1(k)) + J_1^*(f_1(x_1(k), u_1(k)))\}, \tag{2.24}$$

where k is the time step, $x_1(k)$ is the state of FSM_1 at the time step k, U_1 is the input alphabet of FSM_1, $u_1(k)$ is the input of FSM_1 at the time step k, Φ_1 is the transition costs of FSM_1 and f_1 is the transition function of FSM_1. The optimal cost-to-go of DP for FSM_2 is J_2^*, and it is also obtained by (2.20) as follows:

$$J_2^*(x_2(k)) = \min_{u_2(k)\in U_2}\{\Phi_2(x_2(k), u_2(k)) + J_2^*(f_2(x_2(k), u_2(k)))\}, \tag{2.25}$$

where $x_2(k)$, U_2, $u_2(k)$, Φ_2 and f_2 are defined as in (2.24).

Let the optimal cost-to-go of DP for $FSM_1 \times FSM_2$ be J^*. It is obtained by (2.20) as follows:

$$J^*(x(k)) = \min_{u(k)\in U}\{\Phi(x(k), u(k)) + J^*(f(x(k), u(k)))\}, \tag{2.26}$$

where $x(k)$, U, $u(k) = (u_1(k), u_2(k))$, $\Phi = \Phi_1 + \Phi_2$ and f are defined similarly as in (2.24). Equations (2.2) and (2.26) yield

$$J^*(x(k)) = \min_{u(k)\in U}\{\Phi(x(k), u(k)) + J^*(x(k+1))\}. \tag{2.27}$$

We expand (2.27) to step K as follows:

$$\begin{aligned} J^*(x(k)) &= \min_{u(k)\in U}\{\Phi(x(k), u(k)) + \min_{u(k+1)\in U}\{\Phi(x(k+1), u(k+1)) + J^*((x(k+2))\}\}, \\ &= \min_{u(k)\in U}\{\Phi(x(k), u(k)) + \min_{u(k+1)\in U}\{\Phi(x(k+1), u(k+1)) + ... \\ &\quad + \min_{u(K-1)\in U}\{\Phi(x(K-1), u(K-1)) \\ &\quad + \min_{u(K)\in U}\{\Phi(x(K), u(K)) + J^*((x(K+1))\}\}...\}\}, \end{aligned} \tag{2.28}$$

where K is the time horizon of DP. When K is large enough, $J^*((x(K+1)) = 0$. Then, the last line of (2.28) becomes

$$\min_{u(K)\in U}\{\Phi(x(K), u(K)) + J^*((x(K+1))\} = \min_{u(K)\in U}\{\Phi(x(K), u(K))\}. \tag{2.29}$$

Then, (2.17) and (2.29) yield

$$\begin{aligned}\min_{u(K)\in U}\{\Phi(x(K),u(K))\} &= \min_{(u_1(K),u_2(K))\in U}\{\Phi((x_1(K),x_2(K)),(u_1(K),u_2(K)))\},\\ &= \min_{(u_1(K),u_2(K))\in U}\{\Phi_1(x_1(K),u_1(K))+\Phi_2(x_2(K),u_2(K))\}.\end{aligned} \tag{2.30}$$

Since Φ_1 does not depend on u_2 and Φ_2 does not depend on u_1, (2.30) can be rewritten as follows:

$$\min_{u(K)\in U}\{\Phi(x(K),u(K))\} = \min_{u_1(K)\in U_1}\{\Phi_1(x_1(K),u_1(K))\} + \min_{u_2(K)\in U_2}\{\Phi_2(x_2(K),u_2(K))\}. \tag{2.31}$$

When K is large enough, $J_1^*(x_1(K+1)) = 0$ and $J_2^*(x_2(K+1)) = 0$. Then, (2.24), (2.25) and (2.31) imply

$$\min_{u(K)\in U}\{\Phi(x(K),u(K))\} = J_1^*(x_1(K)) + J_2^*(x_2(K)). \tag{2.32}$$

Then, (2.28), (2.29) and (2.32) yield

$$\begin{aligned}J^*(x(k)) = \min_{u(k)\in U}\{&\Phi(x(k),u(k)) + \min_{u(k+1)\in U}\{\Phi(x(k+1),u(k+1)) + ...\\ &+ \min_{u(K-1)\in U}\{\Phi(x(K-1),u(K-1)) + J_1^*(x_1(K)) + J_2^*(x_2(K))\}...\}\}.\end{aligned} \tag{2.33}$$

As in (2.30), $\Phi(x(K-1),u(K-1))$ can be expressed in terms of Φ_1 and Φ_2, so (2.33) becomes

$$\begin{aligned}J^*(x(k)) = \min_{u(k)\in U}\{&\Phi(x(k),u(k)) + \min_{u(k+1)\in U}\{\Phi(x(k+1),u(k+1)) + ...\\ &+ \min_{(u_1(K-1),u_2(K-1))\in U}\{\Phi_1(x_1(K-1),u_1(K-1))\\ &+ J_1^*(x_1(K)) + \Phi_2(x_2(K-1),u_2(K-1)) + J_2^*(x_2(K))\}...\}\}.\end{aligned} \tag{2.34}$$

Similar to (2.31), Φ_1 does not depend on u_2 and Φ_2 does not depend on u_1. Therefore, (2.34) can be rewritten as

$$\begin{aligned} J^*(x(k)) = \min_{u(k)\in U}\{\Phi(x(k),u(k)) + \min_{u(k+1)\in U}\{\Phi(x(k+1),u(k+1)) + ... \\ + \min_{u_1(K-1)\in U_1}\{\Phi_1(x_1(K-1),u_1(K-1)) + J_1^*(x_1(K))\} \\ + \min_{u_2(K-1)\in U_2}\{\Phi_2(x_2(K-1),u_2(K-1)) + \ J_2^*(x_2(K))\}...\}\}. \end{aligned} \tag{2.35}$$

Hence, (2.24), (2.25) and (2.35) yield

$$\begin{aligned} J^*(x(k)) = \min_{u(k)\in U}\{\Phi(x(k),u(k)) + \min_{u(k+1)\in U}\{\Phi(x(k+1),u(k+1)) + ... \\ + J_1^*(x_1(K-1)) + J_2^*(x_2(K-1))\}...\}\}. \end{aligned} \tag{2.36}$$

Repeating (2.33) through (2.36) until step k, we obtain

$$J^*(x(k)) = J_1^*(x_1(k)) + J_2^*(x_2(k)). \tag{2.37}$$

Thus, the optimal cost-to-go for $FSM_1 \times FSM_2$, J^*, is the sum of the optimal cost-to-go for FSM_1 and the optimal cost-to-go for FSM_2. Moreover, since the costs are uncoupled, that is, Φ_1 does not depend on u_2 and Φ_2 does not depend on u_1, J^* is obtained by applying the optimal policies of FSM_1 and FSM_2 to their respective FSMs. By (2.22), the optimal policies of FSM_1 and FSM_2 are u_1^* and u_2^* respectively. Then, together with (2.26), the optimal policy for $FSM_1 \times FSM_2$ is $u^* = \begin{bmatrix} u_1^* \\ u_2^* \end{bmatrix}$.

Claim B.

Assume that $FSM = FSM_1 \times FSM_2$ has cost $\Phi = \Phi_1 + \Phi_2$ and FSM_1 has cost Φ_1. Let $u^* = DP(FSM,\ \Phi)$, $u_1^* = DP(FSM_1,\ \Phi_1)$ and $u_2^* = DP(FSM/FSM_1,\ \Phi - \Phi_1)$. Then, we claim that $u^* = \begin{bmatrix} u_1^* \\ u_2^* \end{bmatrix}$.

Proof B. By definition of pruning (2.8), $FSM/FSM_1 = FSM_2$ because $FSM = FSM_1 \times FSM_2$. Moreover, $\Phi - \Phi_1 = \Phi_2$ by (2.17) and (2.18) because $\Phi = \Phi_1 + \Phi_2$. Then, the optimal policy for FSM/FSM_1 is

$$u_2^* = DP(FSM_2,\ \Phi_2). \tag{2.38}$$

Then, claim B is equivalent to claim A.

2.5 Optimal Policy of Restricted Finite State Machine

Unlike the composed FSMs, the optimal policy of a RFSM may not be the union of the optimal policies of its component FSMs. As previously, without loss of generality, we consider the case of two component FSMs yielding one RFSM. Then, the union of optimal policies of two FSMs may not be the optimal policy of the RFSM.

Consider a ReRFSM as defined in Section 2.2.5. Assume that the transitions in each component FSM of the ReRFSM have associated cost (typically, assigned heuristically) and that the costs are additive (2.17) upon composition (2.13), (2.15) and subtractive (2.18) upon pruning (2.14), (2.16). Input restrictions (2.11) and/or (2.12) are applied during the composition. The optimal policies are obtained by (2.23) after the composition.

Claim C.

Assume that FSM_1 has cost Φ_1 and FSM_2 has cost Φ_2. Assume that $RFSM = FSM_1 \bar{\times} FSM_2$ as described in Sec. 2.2.4

Let $u_1^* = DP(FSM_1, \Phi_1)$, $u_2^* = DP(FSM_2, \Phi_2)$ and $u^* = DP(RFSM, \Phi_1 + \Phi_2)$.

Then, we claim that u^* may not be $\begin{bmatrix} u_1^* \\ u_2^* \end{bmatrix}$.

Proof C. Assume that $u^* = \begin{bmatrix} u_1^* \\ u_2^* \end{bmatrix}$. Assume that $u^*(x(k)) \in u^*$ is the optimal input in state x at step k, and $\begin{bmatrix} u_1^*(x_1(k)) \\ u_2^*(x_2(k)) \end{bmatrix} \in \begin{bmatrix} u_1^* \\ u_2^* \end{bmatrix}$ is the input in state x at step k from union policy. Assume that $\begin{bmatrix} u_1^*(x_1(k)) \\ u_2^*(x_2(k)) \end{bmatrix} \notin U$ as in (2.11). Then, $u^*(x(k)) \neq \begin{bmatrix} u_1^*(x_1(k)) \\ u_2^*(x_2(k)) \end{bmatrix}$, which implies $u^* \neq \begin{bmatrix} u_1^* \\ u_2^* \end{bmatrix}$. Thus, it may be that $u^* \neq \begin{bmatrix} u_1^* \\ u_2^* \end{bmatrix}$.

Thus, the optimal policy of a RFSM is obtained by applying DP directly to the RFSM. Consequently, as we mentioned in the previous section, if the sizes of the state space and the input space of the RFSM are large, obtaining the optimal policy of the RFSM using DP is computationally infeasible.

2.6 Approaches

2.6.1 DP for ReFSM or ReRFSM

The optimal policy of the composed FSM (or RFSM) is computed for every instance when the composed FSM (or RFSM) is updated. The method is called DP for ReFSM (or ReRFSM). In the "normal" mode, the ReFSM (or ReRFSM) operates as a composed FSM (or RFSM) with associated transition cost. DP provides a policy. We use that policy until composition or pruning occurs. Then, the ReFSM (or ReRFSM) and cost are updated and we recompute the policy. The algorithm of DP for ReRFSM is shown in Algorithm 1.

As described in earlier sections, DP for ReFSM is computationally tractable because of the union policy. However, DP for ReRFSM is computationally intractable if the state space is large; unfortunately, many realistic problems are modeled by ReRFSM.

Algorithm 1 DP for ReRFSM

```
1: $\pi^*(k) \leftarrow$ DP for ReRFSM(RFSM($k$), RFSM($k-1$))
2: **if** RFSM($k$) $\neq$ RFSM($k-1$) **then**
3:     $w \leftarrow$ number of component FSMs
4:     **for** $i = 1$ **to** $w$ **do**
5:         $FSM_i \leftarrow$ component FSM
6:         $\Phi_i \leftarrow$ transition cost of $FSM_i$
7:     **end for**
8:     $RFSM \leftarrow FSM_1 \bar{\times} FSM_2 \bar{\times} ... \bar{\times} FSM_w$
9:     $\Phi \leftarrow \Phi_1 + \Phi_2 + ... + \Phi_w$
10:    **for** $i = 1$ **to** $w$ **do**
11:        **if** $FSM_i$ needs to be pruned **then**
12:            $RFSM \leftarrow RFSM \bar{/} FSM_i$
13:            $\Phi \leftarrow \Phi_1 + \Phi_2 + \Phi_{i-1} + \Phi_{i+1} + ... + \Phi_w$
14:        **end if**
15:    **end for**
16:    $\pi^*(k) =$ DP($RFSM$, $\Phi$, $K$)
17: **else**
18:    $\pi^*(k) = \pi(k-1)$
19: **end if**
20: **return** $\pi^*(k)$
```

2.6.2 LBFS for ReRFSM

The motivation of this section is that DP suffers from the curse of dimensionality. Using DP for ReRFSM is computationally expensive, especially if the state space 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 without any prediction, the previous policy is not useful. In such situation, obtaining a policy is useless, and obtaining a solution may be good enough.

Thus, we use a 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:

$$\text{number of next states} \times \text{number of inputs} \leq CC_{ub}. \tag{2.39}$$

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 (size of input alphabet), the number of next states is obtained based on (2.39), 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 algorithms for LBFS for ReRFSM are shown in Algorithm 2.

Algorithm 2 Algorithm of LBFS for ReRFSM

```
$u(k) \leftarrow$ LBFS for ReRFSM(RFSM($k$))
$w \leftarrow$ number of component FSMs
for $i = 1$ to $w$ do
    $FSM_i \leftarrow$ component FSM
    $\Phi_i \leftarrow$ transition cost of $FSM_i$
end for
$RFSM \leftarrow FSM_1 \bar{\times} FSM_2 \bar{\times} ... \bar{\times} FSM_w$
$\Phi \leftarrow \Phi_1 + \Phi_2 + ... + \Phi_w$
for $i = 1$ to $w$ do
    if $FSM_i$ needs to be pruned then
        $RFSM \leftarrow RFSM \; \overline{/} \; FSM_i$
        $\Phi \leftarrow \Phi_1 + \Phi_2 + \Phi_{i-1} + \Phi_{i+1} + ... + \Phi_w$
    end if
end for
$u(k)$ = LBFS($RFSM$, $\Phi$, $fd$, $CC_{ub}$)
return $u(k)$
```

2.7 Optimality of ReFSM and ReRFSM

What the optimal policies for the ReFSM or ReRFSM should be still not clear. The optimal policy for the ReFSM or ReRFSM should generate the optimal cost to-go. Intuitively, DP for ReFSM or ReRFSM seems to generate the optimal policy for the ReFSM or ReRFSM, so yield the optimal cost to-go. DP for ReFSM or ReRFSM generates the local optimal policy for each composed FSM in ReFSM, and each RFSM in ReRFSM; but adding or deleting FSMs from the previous composed FSM or RFSM without any prediction can affect the optimality or ReFSM or ReRFSM; thus, the resulting policy may not be optimal.

However, in the case of DP for ReFSM, the resulting policy tends to stay near the optimal policy because in ReFSM, the optimal policy of the previous composed FSM is connected to the optimal policy of the next composed FSM because the optimal policy of the composed FSM is obtained by the union policy. In other word, they share the same component FSMs and the optimal policy for the same component FSM remains the same. Consequently, if the number of changed FSMs is relatively large, the connections are weaker, and the resulting policy may be far from the optimal policy.

Furthermore, many realistic problems can be modeled by a ReRFSM, but because of the input restrictions, the optimal policies between the RFSMs in the ReRFSM are not well connected. Thus, the policy from DP for ReRFSM may not stay close to the optimal policy.

Consider the simple example FSM as shown in Fig. 2.1. The FSM can be represented as a four-tuple: $X = \{A, B\}$, $U = \{1, 2\}$, $x_0 = A$, $x(k+1) = f(x(k), u(k))$ where f operates as follows: $f(A, 1) = B$, $f(A, 2) = A$, $f(B, 1) = A$, and $f(B, 2) = B$. The goal state is B.

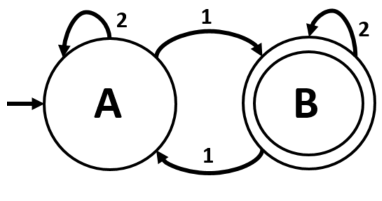

Figure 2.1: Example FSM.

Assume that at step 1, two example FSMs (FSM_1 and FSM_2) are composed with input restrictions to form an RFSM, and the restricted input alphabet (global input restriction) is as follows:

$$U = \{(1,2),(2,1),(2,2)\}. \tag{2.40}$$

In other words, u_1 and u_2 for the input $u = (u_1, u_2) \in U$, cannot be input 1 at the same time. Assume that the transition cost of the FSM_1 is given as $\Phi_1(A,1) = 7$, $\Phi_1(A,2) = 10$, $\Phi_1(B,1) = 10$, and $\Phi_1(B,2) = 0$, and FSM_2 is given as $\Phi_2(A,1) = 5$, $\Phi_2(A,2) = 10$, $\Phi_2(B,1) = 10$, and $\Phi_2(B,2) = 3$.

Assume that $fd = 1$, that is, LBFS behaves as a greedy algorithm. Then, applying DP to the RFSM yields the optimal policy and applying LBFS to the RFSM yields a sub-optimal solution. For DP, the cost to-go from the initial state to the goal state is 22, which is the optimal cost to-go, while it is 25 for LBFS. Both take two steps to reach the goal state.

However, assume that another FSM (FSM_3) is added at step 2 without any prediction. Assume that the transition cost of the FSM_3 is given as $\Phi_3(A,1) = 5$, $\Phi_3(A,2) = 10$, $\Phi_3(B,1) = 10$, and $\Phi_3(B,2) = 0$. Then, at step 2, the three FSMs are composed with input restrictions to form an RFSM, and the restricted input alphabet (global input restriction) is as follows:

$$U = \{(1,2,2),(2,1,2),(2,2,1),(2,2,2)\}, \tag{2.41}$$

and the local input restriction is as follows:

$$f(x,\ (2,2,1)) = \emptyset, \quad \text{if } x \in \{(B,A,A),(B,A,B),(B,B,A),(B,B,B)\}. \tag{2.42}$$

In other words, u_1, u_2, and u_3 for the input $u = (u_1, u_2, u_3) \in U$ cannot be input 1 at the same time, and input $(2,2,1)$ is prohibited at the state (B,A,A), (B,A,B), (B,B,A), and (B,B,B).

Assume that the same parameters are used for the computations. Then, DP for ReRFSM yields a cost to-go from the initial state to the goal state of 83 and it takes five steps to reach the goal state, while LBFS for ReRFSM yields a cost of 43 and it takes three steps; this is, in fact, the optimal cost to-go. LBFS for ReRFSM is much better than DP for ReRFSM because the policy from DP for ReRFSM at step 1 (with two FSMs) leads the states to go (B, A), so at step 2, when FSM_3 is added, the state is (B, A, A) where the optimal policy from the state requires four steps to reach the goal state because of local input restrictions (2.42). Thus, the optimality of ReRFSM may not be achieved by DP for ReRFSM. In other words, current local optimal policy may be useless in the future for ReRFSM.

Apparently, it is not possible to obtain the optimal policy of ReFSM or ReRFSM without knowing the future perfectly. Thus, first, obtaining the optimal policy of RFSM using DP is difficult if the state space is large as described in the previous section, and more importantly, second, obtaining the local optimal policy of RFSM in ReRFSM does not guarantee the optimal policy of ReRFSM. Therefore, heuristics, such as LBFS can be used for ReRFSM.

2.8 Conclusions

Real-world environments are unpredictably dynamic and resources are usually limited, hence, we seek a method that can handle UDEs and limited resource situations. We leverage the facts that FSM are intuitive and easy to use. We defined the RFSM, which is a composed FSM with global and/or local input restrictions that takes into account the resource limitations, and we defined the ReFSM and ReRFSM based on the composition and pruning operations of the FSM to handle UDEs. We proved that the optimal policy of a composed FSM is obtained by union of optimal policies of its component FSMs. Thus, even if the number of component FSMs is large, the optimal policy of the composed FSM can be obtained without computational difficulties. Many real-world problems are formulated by RFSM to take account resource limitations; however, in the case of the RFSM, the union policy is unavailable, and it is computationally difficult to obtain the optimal policy if the number of component FSMs is large.

However, in general, the optimality of the ReFSM or ReRFSM is not achieved by obtaining the local optimal policies of the composed FSMs in ReFSM or RFSMs in ReRFSM in UDEs, where FSMs are composed or pruned in a completely unpredictable manner. Thus, DP for ReFSM or ReRFSM does not guarantee an optimal policy, and it may perform worse than LBFS for ReFSM or ReRFSM. In the case of ReFSM, DP for ReFSM may stay near the optimal policy because the local optimal policies are connected to each other by the union policy, but in the case of ReRFSM, the local optimal policies are not well connected because of input restrictions. Thus, in the case of ReRFSM, the local optimal policy may be useless.

Obtaining the optimal policy of ReRFSM is impossible without knowing the future perfectly, which is seldom the case in reality, and obtaining local optimal policies in ReRFSM may be useless, so using LBFS for ReRFSM in real-world scenarios in UDE and limited resource situations is reasonable.

CHAPTER III

Task Scheduling for Radar Resource Management in Dynamic Environments

3.1 Introduction

In this chapter, we study phased array radar resource management in dynamic environments. By management, we mean task assignment and scheduling. By dynamic environments, we mean that the environment may change unpredictably in time, thus the name UDE. In our scenarios, first, a phased array radar allocates resources in a dynamic environment when the resources are limited; second, multiple phased array radars work together to allocate resources in a dynamic environment when the resources are limited.

We develop a phased array radar scheduler and test it for single radar and multiple radar scenarios. We focus on making decisions of how to allocate finite radar resources when the resources are not sufficient to perform all tasks, in a dynamically changing environment. In other words, our focus is on task scheduling when the set of tasks to be performed changes dynamically and the ability to perform tasks is resource-constrained. Thus, we do not consider all physics of the radar; instead we only consider key physics of the radar that affect decision making. The key radar physics were suggested by radar modeling experts at ONR. Here, we mainly focus on allocation of radar resources to area search and threat tracking for discrimination subject to given constraints, and on implementation of our own original methodology for dynamic modeling and efficiency. By discrimination, we mean collecting information to identify the threat. We use ReRFSM to design a scheduler for the radar and apply DP to obtain the policy, or heuristics to obtain the policy/solution.

3.1.1 Motivation

Our motivating problem is as follows. Consider a geographic area in which two enemy forces operate. The red force consists of multiple threats (used to represent either missiles or aircraft). The blue force consists of ships that carry multi-function phased array radars. The ships are deployed throughout the area; they use their radars to detect threats, and can communicate among themselves. The goal of the red force is to destroy the blue force. The goal of the blue force is to detect and destroy the red force. Our goal is to provide algorithms to detect, discriminate, and obtain as much information as possible on threats, which is then sent to an interceptor unit that processes (attempts to destroy) the threats. This is realistic in view of the standard separation of responsibilities on battleships. We refer to the tasks performed by our algorithms as radar resource management.

Radar resource management belongs to the broad class of task allocation (who does what?) and scheduling (when?) problems. Current radar resource management algorithms are limited to a single radar. Radar resource management is challenging because radar resources are limited; in some scenarios the workloads are severe, and the enemy often attempts to overwhelm the radar.

Our designs do not contain all the physics and details necessary for individual radar control. But they are appropriate for the study of distributed policies/solutions and communication strategies. A distributed system can achieve cooperation of radars using communications. Distribution of tasks across the fleet (to avoid duplication of effort) and communication of information between radars may be advantageous. For distributed fleet-level radar systems, each radar uses its own scheduler rather than one global scheduler. A distributed architecture is advantageous compared to a centralized one from the standpoints of computational time and flexibility of organization.

In addition, the radar resource management must deal with a dynamic environment. The blue force does not know a priori how many threats the red force has at its disposal, or their nature. Every time a new threat is detected, the situation awareness must be updated, and discrimination and tracking are initiated, resource-permitting. Specifically, we consider the following scenarios.

Scenario 1 Consider a phased array radar on a ship, in high seas, surrounded by many threats. The ship is located in a two dimensional geographic area, and all threats are moving toward the radar (or ship) with constant speeds. The number and location of threats are unknown and unpredictable, which makes the environment dynamic and unpredictable. The radar must achieve its goals by allocating a finite amount of resources. The goals of the radar are to ensure zero leakage and to discriminate as many threats as possible. Zero leakage means that the radar should be aware of the threats if they are within a certain range.

Scenario 2 Consider the same battle situation as scenario 1 with multiple ships. Each ship is equipped with one phased array radar that has a finite amount of resources. The goals of the radars are the same as in scenario 1, and the radars have to cooperate to achieve the goals using the finite resources of each radar.

3.1.2 Literature Review

There have been many approaches to model and control of battlespaces. In [22, 34, 64], time-based state space models are used to represent a stochastic system. Discrete system based modeling for adversarial situations can be found in [45, 91] using finite state machines and in [35] using hybrid automata. To represent a discrete decision process, a finite state automaton is used in [50].

A number of papers have studied allocation of multi-function radar resources. Visnevski et al. [102] model the emitter of a multi-function radar based on a generalized semi-Markov process in electronic warfare. Watson and Blair [103] calculate a revisit time to track maneuvering targets using the Interacting Multiple Model. They mix multiple models' states using a Markov process, and estimate the next states to calculate the revisit time. However, they only consider a tracking task even if the radar can perform multiple tasks simultaneously such as searching and tracking. In addition, they do not consider overwhelming situations where the resources are not sufficient to perform all tasks. Blair et al. [13] propose a benchmark problem for highly maneuverable targets. The purpose is to obtain beam pointing control of a phased array radar against highly maneuverable targets in the presence of false alarms and Electronic Counter Measures. They provide a structure of the solutions, so each participant codes his own "tracking algorithm" in the structure to solve the problem to achieve the given performance goal. However, the solution focuses on minimizing tracking error and the targets are already specified for the problem, which is not suitable for realistic battle situations.

Ting et al. [96] develop a dwell scheduling algorithm for a multi-function phased array radar based on a combination of heuristics and scheduling gain in real time. Gosh et al. [44] propose a phased array radar model that does resource allocation and scheduling using a Quality of Service (QoS)-based resource allocation model optimizer and improper nesting of the radar dwells. They allocate the radar resources to each task and also schedule the allocated resources to the radar to meet a jitter requirement. By jitter, we mean a deviation pulse from a digital signal. However, they allocate resources to all tasks, which implies that if there are too many tasks, the radar lowers the utilization bound for the tasks, which causes poor radar performance. Furthermore, none of those radar models are appropriate for multi-radar systems.

Many of multi-function radar tasks scheduling works are based on prioritization. Miranda et al. [69, 67] allocate the radar resources using task scheduling and compare the scheduling algorithms. The scheduling depends on the pre-assigned priority of the tasks allocated to the radar. In [68], the authors develop an adaptive prioritization of the tasks based on a fuzzy-reasoning-based algorithm in dynamically changing tactical environments and compare their design with other prioritization methods. Jiménez et al. [49] model a radar task scheduler using three stages: task priorization, a scheduling algorithm, and temporal planning, and compare different scheduling algorithms. Duron and Proth [29] maximize the number of useful tasks performed, taking into account their priorities. However, the solution depends on the pre-assigned priorities, so the solution may not be optimal. It also depends on probabilities of certain events, which, in practice, are unknown. Therefore, in an overwhelming situation, the radar only performs high priority tasks, which could yield problems such as leakage within a search area because the searching tasks usually have lower priority than the tracking tasks. By leakage, we mean that the radar fails to be aware of the threats when they are in a certain range. Moo [72] develops a method for the scheduling of prioritized tracking and surveillance tasks based on a two-slope benefit function where tracking and high priority surveillance tasks are scheduled first, then lower priority surveillance tasks are scheduled.

Moo and Ding [73] consider a multi-radar configuration, where the networked phased array radars are connected by a communication channel and share the tracking and detection data, and verify a benefit over the independent radars configuration. Severson and Paley [89] describe a distributed multi-radar system. They calculate the optimal position and search radius of the radars to maximize the unified searching area among the radars in a given environment. Also, they allocate the tracking tasks to each radar to balance each radar's utilization. However, there are limitations for dynamic situations, because of assumptions such as the number of targets that the radar can track. These assumptions are not suitable for dynamic environments scenarios because in practice the number of threats is unpredictable.

In addition, many radar control schemes do not consider overwhelming situations, which are important in our case. Many other radar control schemes focus on how to allocate the radar resources to maximize the radar usage while we focus on how to choose the tasks to perform when the radar cannot perform all the tasks because of resource limitations. Tumová et al. [99] consider instances when all of the given specifications cannot be reached simultaneously due to their incompatibility or environmental constraints, which is an overwhelming situation. They find the least violating control in the environment with respect to the given set of mission specifications using a Büchi automaton and a Nested Depth-first Search, which is computationally advantageous compared to exhaustive search. However, unlike [99], we do not have strict behavior rules nor final constraints. Instead, rules are embedded in the cost function, so by minimizing or maximizing the objective function, we obtain a policy/solution.

In [88, 106], we developed a framework for a phased array radar model control using FSM and we applied DP to the FSM. In this chapter, we reformulate the problem using ReRFSM. We also develop a more sophisticated radar scheduler based on our prior work to obtain the policy/solution and test a distributed architecture with multiple radars.

3.1.3 Original Contributions

The original contributions of this chapter are as follows:

1. Radar task scheduler design using ReRFSM: We develop a phased array radar scheduler based on FSM that describes the world with a set of discrete states; the design is amenable to studying radar configurations for fleets. We design mission RFSMs that allow the consideration of limited (radar) resources. RFSMs can

take into account limited resources by restricting some transitions during the composition. We then design ReRFSMs that allow the state space of a mission RFSM to change dynamically; they are suitable to model dynamically changing environments.

2. Policy generation for radar resource management in dynamic environments using DP: We generate policies for a phased array radar system to allocate the radar resources in dynamic environments by applying DP to ReRFSM. The resulting policy allows us to find the radar resource allocation. DP yields a policy for every state, so even though a state may jump to another state unexpectedly, we always have a policy for the resulting state. In other words, if the current decision is interrupted, the policy gives an alternative solution within the state space.

3. Policy or solution generation for radar resource management in dynamic environments using heuristic methods: We generate the polices and solutions for a phased array radar or a multi-radar system to allocate radar resources in dynamic environments even if the radar system encounters an overwhelming situation; Sliding Window Control (SWC), BFS, and (LBFS) are considered. When the number of threats is large, we generate radar resource management policies using SWC and solutions using BFS or LBFS.

4. Distributed multi-radar systems and communication: We develop a distributed architecture for fleet-level radars over communication networks that improves the overall system's performance compared to a decentralized system, without a significant trade-off such as computational time.

To model dynamically changing environments is important because many real world situations are dynamically changing. However, modeling dynamic environments is not easy because it is not possible to perfectly predict dynamically changing environments. Thus, a method is needed to capture dynamically changing environments easily and precisely.

Resource allocation is also important because in practice, resources are limited, so not all desired tasks are performed at the same time step. Resources are used by agents to perform tasks at every time step, and their amount is quantifiable. A multi-function phased array radar has limited resources at a given time step. Sometimes, it is impossible to perform all tasks in the time step because of resource limitations; consequently it is important to choose the tasks to perform first.

Furthermore, in a multi-radar system, cooperation is important because it saves resources, so the radars can perform more tasks. Thus, we present a distributed architecture where the radars can cooperate over a communication channel.

3.2 Radar Modeling

We consider a two dimensional geographic area for the phased array radar model. The radar is physically at the center of a circular disk divided into sectors of equal aperture (Fig. 3.1). A radar can search within a circular area with radius r_{min} and the area is divided into m radar area sectors. Each sector may contain a number of threats. The radar is capable of searching sectors and focusing attention on specific threats, and does so using power. Accordingly, the radar resources are power, and the radar can use a maximum power of P_{total} at each time step. Then, the resources constraint of the radar at each time step k is

$$s_1(k) + s_2(k) + ... + s_m(k) + t_1(k) + t_2(k) + ... + t_n(k) \leq P_{total}, \text{ for all } k, \tag{3.1}$$

where s_i is the transmitted power (P_{trans}) to sector i, m is the total number of radar area sectors, t_j is the transmitted power to threat number j, and n is the total number of threats.

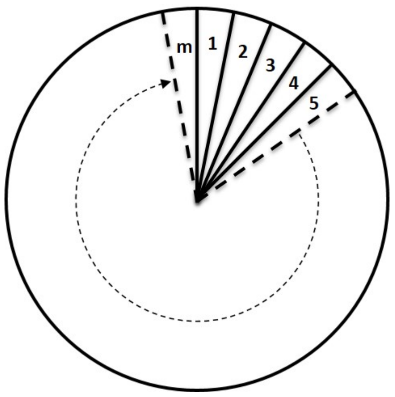

Figure 3.1: Example of radar area sector.

The integers s_i and t_j have the following constraints at each time step k:

$$\begin{aligned} s_i(k) = 0 \text{ or } Bs \leq P_{trans_{max}},\ i = 1, 2, ..., m, \\ 0 \leq t_j(k) \leq P_{trans_{max}},\ j = 1, 2, ..., n, \end{aligned} \tag{3.2}$$

where Bs is the transmitted power required for searching each radar area sector and $P_{trans_{max}}$ is the maximum transmitted power that the radar can use for one task. To ensure zero leakage, the radar should search each sector at least once every maximum revisit time (REV_{max}) to acquire the threat in a certain range. Assume that the scanning rotation rate is fixed. Then, threat acquisition happens when a certain QoS is satisfied for s_i. The QoS for each threat j in sector i at each time step k is given by

$$QoS_j(s_i, r_j, \sigma_j, k) = \frac{Q_0 s_i(k) \sigma_j(k)}{r_j(k)^4}, \tag{3.3}$$

where r_j is the distance between the radar and threat j, σ_j is the radar cross section of threat j, and Q_0 is a normalization constant. Once the desired QoS ($QoS_{desired}$) is determined, which ensures the acquisition of the target in the radar area sector, Bs can be determined as follows. Assume that we know Q_0. If we have the minimum radar range (r_{min}) and the minimum radar cross-section (σ_{min}) of the threat that the radar would acquire at the minimum radar range, we can calculate Bs for $QoS_{desired}$ using the following equation:

$$Bs = \text{ceil}\left(\frac{QoS_{desired} r_{min}^4}{Q_0 \sigma_{min}}\right), \tag{3.4}$$

where the function ceil rounds up to the next integer and $Bs \leq P_{trans_{max}}$ as in (3.2).

Note that s_i should be binary, that is either 0 or Bs, to always satisfy the desired QoS. Then, $QoS_{desired}$ is always satisfied for any threat in a disk of radius r_{min} that has radar cross-section larger than σ_{min}. This allows the radar to ensure the acquisition of the threat for r_{min} and σ_{min}.

To discriminate the threat, we need to track the threat longer than the discrimination time (T_d) with return power (P_{return}) larger than the minimum required return power ($P_{return_{min}}$) at every time step during tracking. The return power for threat j at each time step k is

$$P_{return_j}(k) = \frac{t_j(k) K_G \sigma_j(k)}{r_j(k)^4}, \tag{3.5}$$

where K_G is the radar constant. Assume that we are given $P_{return_{min}}$. Then, we can compute the required transmitted power to the threat to satisfy $P_{return_{min}}$ as function of r:

$$t_{req}(r) = \frac{P_{return_{min}} r^4}{K_G \sigma}, \tag{3.6}$$

where $t_{req}(r) \leq P_{trans_{max}}$ as in (3.2). The parameters for the radar model are summarized in Table 3.1.

Table 3.1: Radar model parameters

P_{total}	Maximum power that the radar can use at each time step
$P_{trans_{max}}$	Maximum transmitted power that the radar can use for one task
$P_{return_{min}}$	Minimum required return power for the radar to discriminate targets
B_s	Required power for searching each radar area sector
Q_0	Normalization constant for the radar
K_G	Radar constant
$QoS_{desired}$	Desired QoS
r_{min}	Minimum radar range that the radar should search
σ_{min}	Minimum radar cross section of the target that the radar should find
REV_{max}	Maximum revisit time for the sector
T_d	Discrimination time for the target

3.3 Problem Formulation

The problem is formulated as follows. Assume that the phased array radar (or radars) is (are) located at the center of a Manhattan grid along which all threats move. The radar has an amount of resources, P_{total} at each time step k as in (3.1) and the radar can use a maximum of $P_{trans_{max}}$ for each threat or sector. The geographic area is divided into m radar area sectors as in Fig. 3.1. Each sector has a revisit deadline, REV_{max} (3.8), and desired QoS $QoS_{desired}$. The radar has constant K_G, normalization constant Q_0, minimum distance to search r_{min}, and minimum cross section to search σ_{min}. There are nt threats moving toward the radar with different constant speeds from different locations. The minimum required return power for tracking is $P_{return_{min}}$ and the time required to discriminate threats is T_d. Given the above data, we want to find a policy/solution for the radar to allocate its resources by minimizing the cumulative cost.

3.4 Task Scheduler Design

The assumptions are: (i) Events are discrete, i.e., events happen slowly enough relative to the time constants of the radar that we can eliminate monitoring of continuous time variables. This is the basis for employing logic-level models (FSMs). (ii) We consider a two dimensional geographic area. (iii) We assume that the radar is not allowed to have resources left at any time step if there is any task left. (iv) We assume perfect communications between the radars of the multi-radar system. By perfect communication, we mean no information loss, no delay and no fail.

3.4.1 Sector Finite State Machine

We keep track of the revisit time for each sector i. The revisit time for each sector i at time step k ($REV_i(k)$) satisfies

$$\begin{aligned} REV_i(k+1) &= REV_i(k)+1, \text{ if } s_i(k)=0, \\ REV_i(k+1) &= 0, \text{ if } s_i(k)>0. \end{aligned} \tag{3.7}$$

Then, the constraint to ensure zero leakage is:

$$REV_i(k) \leq REV_{max}, \text{ for all } k, \tag{3.8}$$

where REV_{max} is the maximum revisit time for all sectors that the radar should search before the sectors reach REV_{max}. This constraint makes the radar search each sector at least once every REV_{max} time steps. We want to discriminate as many threats as possible with respect to the constraints described above.

We use the formalism of FSMs to model the radar area sectors. Fig. 3.2 shows the FSM for a radar area sector when the revisit deadline (REV_{max}) is four. Thus, Bs means the sector is searched and 0 means the sector is not searched. Each sector only allows four time steps of not being searched. Wherever the state is, if the input is Bs, REV is reset to zero.

A Sector FSM is a four-tuple (no output alphabet and output function). The state space is $X = \{SEARCH, NO\ SEARCH1, NO\ SEARCH2, NO\ SEARCH3, NO\ SEARCH4\}$. The input alphabet is $U = \{0,\ Bs\}$. The initial state is $x_0 =$

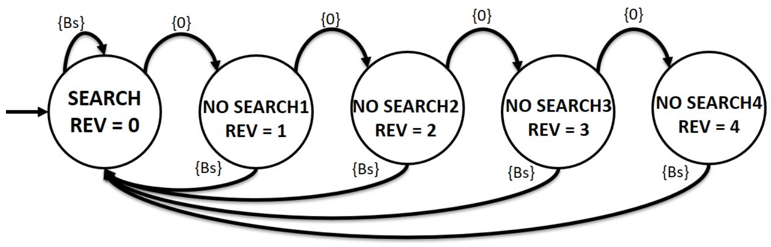

Figure 3.2: Sector FSM when $REV_{max} = 4$.

$SEARCH$. The next state function is f_s:

$$x_s(k+1) = f_s(x_s(k), s), \tag{3.9}$$

where x_s is the state of each radar sector for searching and f_s is defined in Fig. 3.2.

3.4.2 Threat Finite State Machine

As we stated in Section 3.2, to discriminate the targets, we need to keep a record of tracking time for each threat. The consecutive Tracking Time for each threat j at time step k ($TT_j(k)$) always starts from zero ($TT_j(0) = 0$), and satisfies:

$$\begin{aligned} TT_j(k+1) = TT_j(k) + 1,\ if\ P_{return_j}(k) \geq P_{return_{min}}, \\ TT_j(k+1) = 0,\ if\ P_{return_j}(k) < P_{return_{min}}. \end{aligned} \tag{3.10}$$

As we describe in Section 3.2, t_{req} guarantees $P_{return} \geq P_{return_{min}}$. The reward for discrimination of the threat j, R_j, at each time step k is,

$$\begin{aligned} R_j(k) = 1,\ \text{if}\ TT_j(k) \geq T_d, \\ R_j(k) = 0,\ \text{otherwise}. \end{aligned} \tag{3.11}$$

Once $R_j = 1$, we do not evaluate the reward for threat j again. Fig. 3.3 shows an FSM for tracking time for threats when the discrimination time (T_d) is four. Thus, t_{req} means the threat is tracked, 0 means the threat is not tracked and Rew is the set of real numbers where the reward (R) resides. Each threat is discriminated after being tracked for four consecutive time steps. Wherever the state is, if the input is 0, TT is reset to zero except in the last state because the threat is already discriminated.

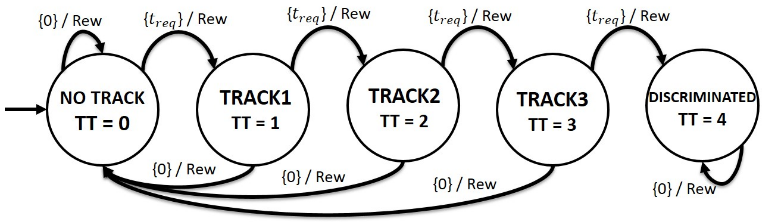

Figure 3.3: Threat FSM when $T_d = 4$.

Each detected threat FSM can be represented by a six-tuple. The state space is $X = \{NO\ TRACK,\ TRACK1,\ TRACK2,\ TRACK3,\ DISCRIMINATED\}$. The input alphabet is $U = \{0,\ t_{req}\}$ and the output alphabet is $Y = Rew$, where Rew is the set of real numbers where the reward (R) resides. The initial state is $x_0 = NO\ TRACK$. The state transitions are governed by:

$$x_t(k+1) = f_t(x_t(k), t), \tag{3.12}$$

where x_t is the state of each detected threat for tracking, f_t is defined in Fig. 3.3 and t is the transmitted power assigned to the threat. The output is:

$$y_t = g_t(x_t(k), t), \tag{3.13}$$

where y_t is the output and g_t is defined in Fig. 3.3.

3.4.3 Mission Restricted Finite State Machine

By composing the sector FSM and the threat FSM, we obtain an RFSM for the mission. Assume that we have m sectors and n threats. Then, during the composition, the input alphabet of the mission RFSM, U, is defined as follows:

$$U = \{u \in U_1 \times U_2 \times \ \dots \ \times U_n \times U_{n+1} \times U_{n+2} \ \dots \ \times U_{n+m} : sum_P(u) \leq P_{total}\}, \tag{3.14}$$

where $sum_P(u)$ means sum of the required powers to perform the input u. The input alphabet is restricted by (3.1).

The representations of sector FSM and threat FSM are convenient because composition of these FSMs allows the representation of all states of the system. Thus, DP can explore every situation and find the optimal policy for all possible situations. In other word, the optimal policy knows what it should do at every state.

3.4.4 Cooperation

For the distributed system with communication, consider the formation of three phased array radars shown in Fig. 3.4. Each radar has the same r_{min} and the same $m = 4$. The areas monitored by the individual radars overlap. We set a threshold such that if one radar sub-area overlaps with at least a certain percentage (say, 90%) of another radar sub-area, we assume that the two sub-areas are the same. Under this assumption, we can say that sub-areas 4A and 2B are the same, as are sub-areas 4B and 2C. These sub-areas are defined as redundant sub-areas.

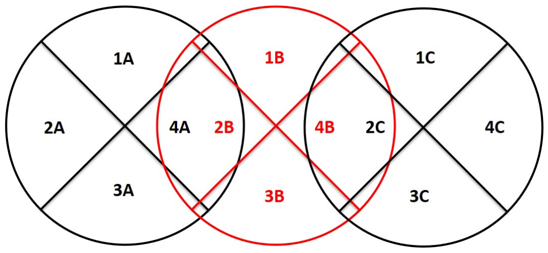

Figure 3.4: Multi radar sectors example.

We introduce a distributed architecture over a communication channel as follows. The radars share the revisit times of redundant sub-areas. For example, in Fig. 3.4, the first radar and second radar share the revisit time for sub-area 4A = 2B, and the second radar and third radar share the revisit time for sub-area 4B = 2C. Thus, if the first radar searches the sub-area 4A, the state of both mission RFSMs of the first and second radars are updated accordingly. In other words, sharing the revisit times of redundant sub-areas can update the states of the relevant radars. Then, even if the whole system is not centralized, this system architecture gives better radar area search performance than the decentralized system because in the decentralized case, individual radars only implement their own policy/solution without regard for the policies/solutions of other radars. Hence, in the decentralized architecture, redundant sub-areas may be searched more frequently than other sub-areas, where the distributed architecture searches evenly. In addition, all the radars share information about discriminated threats. Thus, the states of the mission RFSMs of the relevant radars are updated based on the information. As a consequence, the system does not track a threat if it has been discriminated by one of the radars. Furthermore, the tasks are assigned to the idlest radar first, so the radars that have fewer tasks than the neighboring radars are able to help the neighboring radars by searching the redundant sub-areas. This allows radars to save resources and improves the overall system performance.

3.5 Policy and Solution Approach

Given a ReRFSM, different methods can be applied to obtain the policy and the solution. Each component of the ReRFSM is assigned a transition cost. The transition cost is assigned by a heuristic using REV, TT, r and v of the threat at the time when the FSMs are created. The transition cost of the ReRFSM is the sum of the transition costs of its components as described in Sec. 2.2.6. When composition occurs, the transition cost of the new component is added to the transition cost of the ReRFSM. When pruning occurs, the transition cost of the pruned FSM is subtracted from the transition cost of the ReRFSM.

3.5.1 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 as shown in Sec. 2.4. However, for RFSM, this does not hold due to the restricted inputs in RFSM as shown in Sec. 2.5. Thus, the optimal policy of RFSM has to be obtained by applying DP directly to the whole RFSM.

ReRFSMs allow the state space of an FSM to change dynamically as defined in Section 2.2.5. For example, in our scenario, when the radar detects new enemies, threat FSMs are created and composition occurs. If a threat is discriminated, the FSM of that threat reaches an absorbing state and pruning occurs when needed. The algorithm of DP for ReRFSM for the radar task schedule is shown in Algorithm 3.

Algorithm 3 DP for ReRFSM

```
1: $\pi^*(k) \leftarrow$ DPforReRFSM$(nt(k), nt(k-1))$
2: **if** $nt(k) \neq nt(k-1)$ **then**
3:     **for** $i = 1$ **to** $m$ **do**
4:         $FSMs_i \leftarrow$ sector FSM of sector $i$
5:         $\Phi_{s_i} \leftarrow$ transition cost of $FSM_{s_i}$
6:     **end for**
7:     **for** $j = 1$ **to** $n$ **do**
8:         $FSM_{t_j} \leftarrow$ threat FSM of threat $j$
9:         $\Phi_{t_j} \leftarrow$ transition cost of $FSM_{t_j}$
10:    **end for**
11:    $RFSM_{sys} \leftarrow FSM_{s_1} \bar{\times} FSM_{s_2} \bar{\times} ... \bar{\times} FSM_{s_m} \bar{\times} FSM_{t_1} \bar{\times} FSM_{t_2} \bar{\times} ... \bar{\times} FSM_{t_n}$
12:    **for** $j = 1$ **to** $n$ **do**
13:        **if** $FSM_{t_j}$ needs to be pruned **then**
14:            $RFSM_{sys} \leftarrow RFSM_{sys} \bar{/} FSM_{t_j}$
15:        **end if**
16:    **end for**
17:    $\pi^*(k) = \mathrm{DP}(RFSM_{sys}, \Phi, K)$
18: **else**
19:    $\pi^*(k) = \pi(k-1)$
20: **end if**
21: **return** $\pi^*(k)$
```

3.5.2 Heuristics for ReRFSM

The motivation for this section is that DP suffers from the curse of dimensionality. The first idea is to only use the inputs that use more than a certain threshold of power, P_{lb}, because in an overwhelming situation, using as many resources as possible can take many tasks. Furthermore, we consider the following methods.

Algorithm 4 SWC for ReRFSM

```
$\pi(k) \leftarrow$ SWCforReRFSM($nt(k), nt(k-1), window(k), window(k-1)$)
if $nt(k) \neq nt(k-1)$ then
    if $n > q$ then
        $nt_{win}(k) \leftarrow$ sort($nt(k)$)
        $\bar{n} \leftarrow q$
    else
        $nt_{win}(k) \leftarrow nt(k)$
        $\bar{n} \leftarrow n$
    end if
end if
if $nt(k) \neq nt(k-1)$ or $window(k) \neq window(k-1)$ then
    for $i = 1$ to $m$ do
        $FSMs_i \leftarrow$ sector FSM of sector $i$
        $\Phi_{s_i} \leftarrow$ transition cost of $FSM_{s_i}$
    end for
    for $j = 1$ to $\bar{n}$ do
        $FSM_{t_j} \leftarrow$ threat FSM of threat $j$ in $nt_{win}(k)$
        $\Phi_{t_j} \leftarrow$ transition cost of $FSM_{t_j}$
    end for
    $RFSM_{sys} \leftarrow FSM_{s_1} \bar{\times} FSM_{s_2} \bar{\times} ... \bar{\times} FSM_{s_m} \bar{\times} FSM_{t_1} \bar{\times} FSM_{t_2} \bar{\times} ... \bar{\times} FSM_{t_{\bar{n}}}$
    for $j = 1$ to $\bar{n}$ do
        if $FSM_{t_j}$ needs to be pruned then
            $RFSM_{sys} \leftarrow RFSM_{sys} \overline{/} FSM_{t_j}$
        end if
    end for
    $\pi(k) =$ DP($RFSM_{sys}$, $\Phi$, $K$)
    $window(k) \leftarrow$ update($nt_{win}(k), \pi(k)$)
else
    $\pi(k) = \pi(k-1)$
    $window(k) \leftarrow$ update($nt_{win}(k), \pi(k)$)
end if
return $\pi(k)$, $window(k)$
```

3.5.2.1 Sliding Window Control for ReRFSM

We only consider urgent threats when the number of threats is large. When the number of threats is large, it is computationally impossible to generate the policy because the state space and input space are extremely large. Thus, in this situation, the scheduler sorts the threats by predefined priority, and generates a policy for the window that contains the first q highest priority threats. As soon as at least one of the threats is taken care of, the window is slid over to generate the next policy. With this method, many threats can be taken care of without computational complexity. We call this a SWC.

In SWC, the number of detected threats is evaluated and if the number exceeds the threshold, q, SWC generates a sliding window of threats by priority. Then, DP is applied within the window to obtained the policy for the window. The algorithm of SWC for ReRFSM for the radar task schedule is shown in Algorithm 4.

3.5.2.2 Breadth-First Search and Limited Breadth-First Search for ReRFSM

Using DP for ReRFSM is computationally expensive even if we use SWC, especially if REV_{max} and T_d are large. In addition to that, if the environment is rapidly changing, for example, the number of detected threats is rapidly changing, the ReRFSM has to be recomposed frequently, so the previous policy is useless as described in Sec. 2.7. In such situations, we may not need DP for ReRFSM for the policy; instead we may want to find the solution for the current state. We may lose robustness of the policy, but if the solution is obtained fast enough, we can quickly generate the solution for any state. Thus, we use BFS or LBFS algorithm with fixed depth (fd) to get the solution for the current state as we described in Sec. 2.6.2. The algorithms for BFS or LBFS for ReRFSM are similar to Algorithm 3.

3.6 Scheduler Architecture

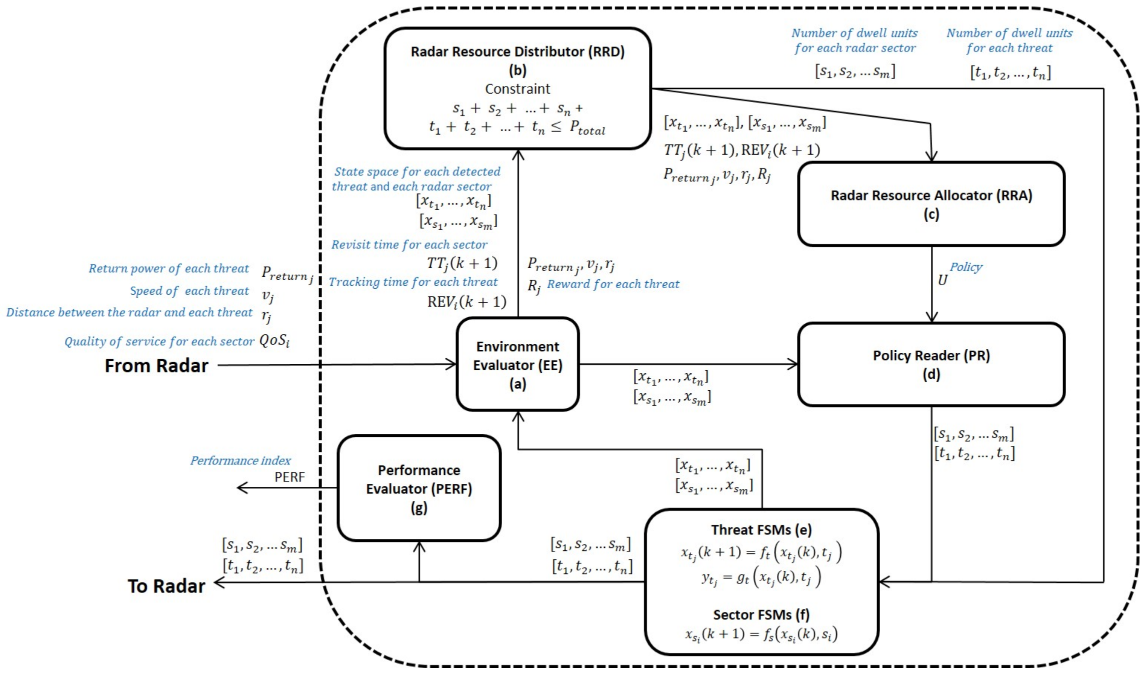
Radar Resource Distributor (RRD) (b)
Constraint
$s_1 + s_2 + \ldots + s_n + t_1 + t_2 + \ldots + t_n \le P_{total}$
Number of dwell units for each radar sector $[s_1, s_2, \ldots s_m]$
Number of dwell units for each threat $[t_1, t_2, \ldots, t_n]$
$[x_{t_1}, \ldots, x_{t_n}], [x_{s_1}, \ldots, x_{s_m}]$
$TT_j(k+1), \mathrm{REV}_i(k+1)$
$P_{return\,j}, v_j, r_j, R_j$
Radar Resource Allocator (RRA) (c)
Policy U
State space for each detected threat and each radar sector $[x_{t_1}, \ldots, x_{t_n}]$ $[x_{s_1}, \ldots, x_{s_m}]$
Revisit time for each sector $TT_j(k+1)$
Tracking time for each threat $\mathrm{REV}_i(k+1)$
$P_{return\,j}, v_j, r_j$
R_j Reward for each threat
Return power of each threat $P_{return\,j}$
Speed of each threat v_j
Distance between the radar and each threat r_j
Quality of service for each sector QoS_i
From Radar
Environment Evaluator (EE) (a)
$[x_{t_1}, \ldots, x_{t_n}]$ $[x_{s_1}, \ldots, x_{s_m}]$
Policy Reader (PR) (d)
$[s_1, s_2, \ldots s_m]$ $[t_1, t_2, \ldots, t_n]$
Performance index PERF
Performance Evaluator (PERF) (g)
$[x_{t_1}, \ldots, x_{t_n}]$ $[x_{s_1}, \ldots, x_{s_m}]$
$[s_1, s_2, \ldots s_m]$ $[t_1, t_2, \ldots, t_n]$
To Radar
Threat FSMs (e)
$x_{t_j}(k+1) = f_t(x_{t_j}(k), t_j)$
$y_{t_j} = g_t(x_{t_j}(k), t_j)$
Sector FSMs (f)
$x_{s_i}(k+1) = f_s(x_{s_i}(k), s_i)$

Figure 3.5: Scheduler diagram.

In this section, we describe the scheduler architecture used to perform radar resource management. The scheduler consists of subsystems (a) through (h), as shown in Fig. 3.5. The inputs of the scheduler from the radar are return power, speed of detected threats, distance between detected threats and the radar, and QoS. The outputs to the radar are the assigned transmitted power for each threat and sector. The descriptions of and relationships between the subsystems are as follows.

(a) Environment Evaluator (EE): EE evaluates environment situations based on the inputs from the radar, such as number of detected threats, to decide whether the system needs to generate a new policy/solution or use the previous policy/solution. If a new policy/solution is needed, go to (b): otherwise, go to (d).

(b) Radar Resource Distributor (RRD): Assigns transmitted power to all the sectors and all the threats. If there is not enough available power to do this assignment, go to (c): otherwise, go to (e) and (f).

(c) Radar Resource Allocator (RRA): Use DP for ReRFSM or SWC for ReRFSM to obtain a policy, or use BFS for ReRFSM or LBFS for ReRFSM to obtain solution.

Note that in the above sequence of actions, we use DP only when we reach (c) in the sequence. In addition, if the numbers of radar sectors and threats are large, we use heuristics in RRA to reduce the computation time.

(d) Policy Reader (PR): PR interprets the policy/solution generated by RRA in terms of radar resources for each sector and threat, then gives the result to the radar.

(e) Threat FSMs: Threat FSM takes assigned transmitted power for each threat as inputs and updates the states.

(f) Sector FSMs: Sector FSM takes assigned transmitted power for each radar sector for searching as inputs and updates the state.

(g) Performance Evaluator (PERF): PERF generates an index to evaluate radar performance easily.

The joint operation of the radar and scheduler can be summarized as follows. It starts from the RRD by receiving inputs from the EE. Then, the system assigns transmitted power to all tasks and evaluates the maximum power constraint. If the power constraint (3.1) is not violated, the system yields the final resource allocation result for the radar. If the constraint is violated, the system moves to RRA. Then, RRA yields the resource allocation policy/solution using one of the approaches. Then, the PR interprets the policy/solution into resource allocation results for the radar inputs according to the current state and saves the policy. The radar takes the inputs from the scheduler and returns information on the radar area sectors and detected threats to the EE. Then the EE decides whether the system needs to generate a new policy/solution or use the current one. After all inputs for the radar are generated, PERF yields a performance index to evaluate the performance of the inputs.

3.7 Simulations and Results

We simulate two scenarios as described in Section 3.1.1: a one radar case, and a three-radar case. For both scenarios, the goal of the radar(s) is to detect and discriminate the threats. Once the threats are discriminated, information about them is sent to an interceptor unit that destroys them. For the one radar case, we compare four methods for ReRFSM for a small scale simulation: DP, SWC, BFS, and LBFS, to find out whether using heuristics generates as good a policy/solution as using DP. We then use SWC and LBFS for a large scale simulation. For the three-radar case, we only use LBFS but we compare centralized and distributed architectures. To compare each method, we use performance metrics as defined in Table 3.2.

Table 3.2: Performance metrics

Metrics	**Descriptions**
$time_{max}$	Maximum computation time: the longest time for DP, SWC, BFS, or LBFS to generate the policy/solution.
$FAIL$	Number of fail cases where the Red force wins, that is, a threat reaches the radar before detection or discrimination.
REV_{avg}	Average revisit time of all sectors: a large value means that the radar does not search the sectors often and evenly, so a small value is desired.
REV_{high}	Highest revisit time of the sectors: a smaller value than REV_{max} is desired.
TI_{avg}	Average time to impact when the threat is discriminated: a small value means that the radar discriminates the threats late, so a large value is desired.
TI_{low}	Minimum time to impact when the threat is discriminated: a small value means that the radar discriminates the threats late, so a large value is desired.

3.7.1 One radar case: small scale

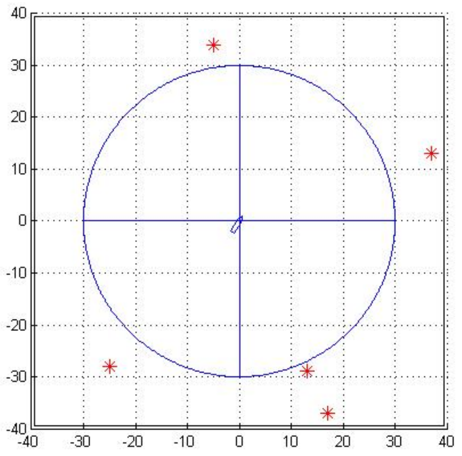

Figure 3.6: Snapshot of one radar scenario.

We first run small scale simulations that the radar handles easily to compare the solution approaches (DP, SWC, BFS, and LBFS) and their simulation times. A snapshot of a one radar scenario is shown in Fig. 3.6. The ship and radar are indicated in blue, and threats are indicated by red stars. For the simulations, we set the variables as shown in Table 3.3.

Table 3.3: One radar scenario simulation variables

P_{total}	200	Q_0	15,000
$P_{trans_{max}}$	64	r_{min}	30
m	4	σ_{min}	2
REV_{max}	2	nt	5
$QoS_{desired}$	2	T_d	2
K_G	20,000	P_{lb}	140
fd	4	CC_{ub}	35,000
$P_{return_{min}}$	5		

We set priorities for our sliding window by using time to impact as a metric, and set a maximum of three threats as a threshold for SWC. We run 30 simulations using the same variables but with different initial positions and speeds for each threat. The average performance metrics for the 30 simulations are shown in Table 3.4.

Table 3.4: Average performance metrics for small scale simulations for one radar

	DP	**SWC**	**BFS**	**LBFS**
$time_{max}$	7.74	3.82	0.60	0.44
$FAIL$	0	0	0	0
REV_{avg}	0.36	0.36	0.33	0.33
REV_{high}	2	2	2	2
TI_{avg}	12.39	12.41	12.53	12.53
TI_{low}	5.22	5.23	5.32	5.31

In terms of maximum computation time, the LBFS is the best, followed by BFS, SWC and DP. The maximum computation time for the LBFS is only 0.44 seconds while for DP it is 7.74 seconds, a factor of 18 times.

No method encounters a fail case. The other performance metrics are not significantly different between each method. BFS and LBFS yield slightly better performance metrics (smaller REV_{avg} and larger TI_{avg} and TI_{low}) than DP and SWC. It is because the BFS and LBFS update ReRFSM at every time step, which allows them to use the latest information on the threats, while DP and SWC only update ReRFSM when a new threat FSM is added because the policy can be used before a new threat FSM arises. As we mentioned earlier, if the environment changes are frequent, we may not need the full policy; instead, we need the solution for the current state, as the simulation results suggest.

Therefore, we can conclude that the LBFS performs as well as DP on the metrics that were evaluated, and significantly reduces computation time. In addition, LBFS yields better decisions as DP, so we can save much computation time by using LBFS. This suggests that LBFS may be a good method for large scale problems.

3.7.2 One radar case: large scale

We then simulate larger scale problems that the radar does not easily handle by increasing nt, REV_{max} and T_d. For the large scale simulations, we set the variables as shown in Table 3.5.

Table 3.5: One radar scenario simulation variables

P_{total}	200	Q_0	15,000
$P_{trans_{max}}$	64	r_{min}	30
m	4	σ_{min}	2
REV_{max}	3	nt	20
$QoS_{desired}$	2	T_d	4
K_G	20,000	P_{lb}	140
fd	4	CC_{ub}	35,000
$P_{return_{min}}$	5		

In addition, we set priorities for our sliding window by using time to impact as a metric, and set a maximum of three threats as a threshold for SWC. We run 10 simulations using the same variables but where initial positions and speeds for each threat are different. The average performance metrics of the 10 simulation results are shown in Table 3.6.

Table 3.6: Average performance metrics for large scale simulations for one radar

	SWC	**LBFS**
$time_{max}$	465.73 [s]	8.05 [s]
$FAIL$	0.5	0.1
REV_{avg}	0.58	0.83
REV_{high}	3	3
TI_{avg}	7.63	10.48
TI_{low}	0.88	1.81

The maximum computation time for LBFS is much faster than SWC, around 58 times faster. Because the number of threats is large for our configuration (20), there are some fail cases. Consequently, TI_{low} is small, which means that the radar discriminates some threats at very close range to the radar. However, TI_{avg} for LBFS is 10.48, which means that the radar still discriminates most threats far enough from the radar if we compare this to the small scale problem (where the range was 12.39 - 12.53).

The main difference between SWC and LBFS is that the SWC yields better performance for REV_{avg} while LBFS yields better performance for TI_{avg} and TI_{low}. It is because the SWC can only handle three threats at each time step, so SWC uses the remaining resources for sector searches. Meanwhile, LBFS uses more resources for threat tracking as far as the sectors do not reach the maximum revisit time. Consequently, LBFS can handle more threats than SWC and ensure revisit deadlines for the sectors, so LBFS generates fewer fail cases, that is, LBFS outperforms SWC.

The results indicate that the radar can handle many threats using LBFS but may fail on some threats because of limited resources.

3.7.3 Three radar case: large scale

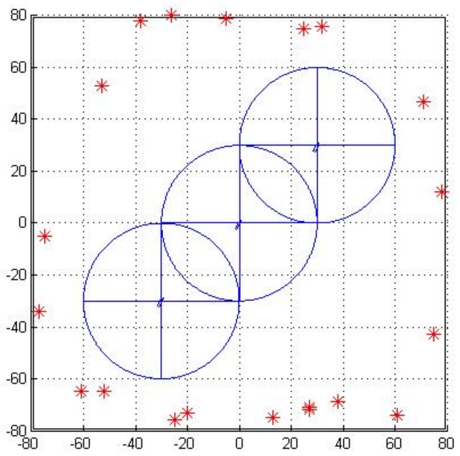

Figure 3.7: Snapshot of three radars scenario.

For a multi-radar system, we compare a decentralized system (without communication) and a distributed architecture (with communication) as described in Section 3.4.4. Both systems are built based on LBFS. A snapshot of a three radar scenario is shown in Fig. 3.7. Ships and radars are indicated in blue, and the threats are indicated by red stars. For the three radar scenario, we set the variables as shown in Table 3.7.

Table 3.7: Three radar scenario simulation variables

P_{total}	200	Q_0	15,000
$P_{trans_{max}}$	64	r_{min}	30
m	4	σ_{min}	2
REV_{max}	3	nt	20
$QoS_{desired}$	2	T_d	3
K_G	20,000	P_{lb}	140
fd	4	CC_{ub}	35,000
$P_{return_{min}}$	5		

We run 10 simulations using the same variables, but with different initial positions and speeds for each threat. The average performance metrics for the 10 simulation runs are shown in Table 3.8.

Table 3.8: Average performance metrics for three radar case

	Decentralized LBFS	**Distributed LBFS**
$FAIL$	0.30 / 0 / 0.40	0 / 0 / 0
REV_{avg}	0.32 / 0.56 / 0.34	0.29 / 0.25 / 0.29
REV_{high}	3 / 3 / 3	3 / 3 / 3
TI_{avg}	2.10 / 9.09 / 3.02	2.21 / 10.31 / 3.28
TI_{low}	0.86 / 6.23 / 1.17	0.86 / 7.52 / 1.35

We observe that in the distributed architecture, radars mostly have better performance metrics than in the decentralized system. TI_{avg} values for the distributed architecture are improved over those of the decentralized system. It is because the radars in the distributed architecture share information about discriminated threats, so as soon as the radar detects one of the threats discriminated by the other radars, the radar gets time to impact index, TI. Also, the radars in the distributed architecture have smaller REV_{avg} and REV_{high}. This means that, in the distributed architecture, the second radar helps the first and third radars by searching redundant sub-areas more frequently because they share revisit times of redundant sub-areas, so the first and third radars can use more resources to track threats. The distributed system is very useful because only simple communications (sharing only revisit time and information on discriminated threats) improve the overall performance a great deal. A centralized system [106] should have better results than the distributed architecture. However, it is nearly impossible to get a solution because of the curse of dimensionality of SSM, even in small scale problems. Thus, in practice, the distributed architecture is a good method to implement radar resource management with proper choice of communications.

3.8 Conclusions

In this chapter, we design a task scheduler for a multi-function phased array radar using ReRFSMs for radar resource allocation in dynamically changing environments. We design ReRFSMs that allow the state space of an FSM to change dynamically. FSMs allow us to model the radar scheduler conveniently because they are intuitive, easy to use and amenable to composition and pruning operations and analysis. RFSM allows one to design a composed FSM in the presence of resource limitations by restricting some of the inputs. We develop a phased array radar resource allocation algorithm using DP for ReRFSM that handles situations where the set of tasks to be performed changes dynamically and the ability to perform tasks is resource-constrained. DP yields a policy for every state, so we always have alternative decisions within the state space although the states may change unexpectedly. We also design heuristic methods; SWC for ReRFSM, BFS for ReRFSM, and LBFS for ReRFSM, and implement them in the scheduler for large numbers of threats and large numbers for REV_{max} and T_d. Our results show that using DP performs well but takes more computation time compared to using the heuristics. However, there is no significant difference in the radar performance, so our approach using LBFS is effective. The resource allocation results depend on a cost function that is based on a heuristic; the results are intuitive and acceptable for real-world scenarios. We also develop a distributed architecture using communication for fleet-level radar systems. The distributed architecture performs better than the decentralized one, as shown by the facts that the distributed architecture has better overall performance metrics than the decentralized one, and that the distributed architecture can handle more threats than the decentralized one in the same battle situation.

CHAPTER IV

Unpredictably Dynamic Environment Patrolling

4.1 Introduction

4.1.1 Motivation

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 [86]. 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: an 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. In short, we allow for UDEs. If an 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 [86].

Consider the following UAV patrolling mission scenario: An 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. 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 an UAV mission planner that can handle such UDE patrolling scenarios. We design a mission planner for the UAV patrolling problem in UDE using ReRFSM. ReRFSM can handle UDE by allowing the state space of a FSM to change dynamically to follow the environment. ReRFSM can represent the problem by prohibiting some of the transitions during the composition. By applying DP to ReRFSM, a robust policy is obtained and by applying a heuristic method, LBFS, to ReRFSM, a solution is obtained. 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 (off-line) computation; the method is thus applicable to small problem instances. For large problem instances, LBFS quickly generates a solution for the current state.

4.1.2 Literature Review

Autonomous vehicle patrolling has been studied at great length in the literature. In [5], 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. In [38], teamed UAVs perform surveillance near a base relying on Unattended Ground Sensor (UGS) 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 [33], 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 chapter is similar to the TSP or 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 [57]. An algorithm that transforms the heterogeneous, multiple depot, multiple UAVs routing problem to the Asymmetric Traveling Salesman Problem (ATSP) is proposed in [76], so the well known LK heuristic can be applied to solve the problem. In [85], the authors solve the TSP for Dubins' vehicles, which are a standard kinematic model abstraction for UAVs, where the vehicles are assumed to operate in a 2D plane and have turn rate constraints, so that the paths between points may be straight lines or arcs of circles.

The VRP originated in a truck dispatching problem in [24]. 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 [19], 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 [56]. Minimizing emissions and fuel consumption are considered in [40].

Dynamic versions of the VRP are proposed in [78] (DVRP) and in [12] (DTRP). They consider dynamically changing environments. In [84], the UAV is required to visit a dynamically growing set of targets while traveling 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 [31] for multiple UAVs. In [14], 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 [37, 35, 36].

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 chapter, we aim to develop an 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.

4.1.3 Original Contributions

The original contributions of this chapter 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. Our ReRFSM design can handle UDE by allowing the state space to change vigorously to accommodate the changing environment.

2. We generate mission planning policies for an UAV in UDE using DP for ReRFSM. Using DP, we generate policies for every state, allowing optimal and robust mission planning in UDE.

3. We generate mission planning solutions for an UAV in UDE using a heuristic, LBFS for ReRFSM. Using LBFS, we generate a solution 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.

4.2 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.

4.2.1 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). Fig. 4.1 shows the example scenario.

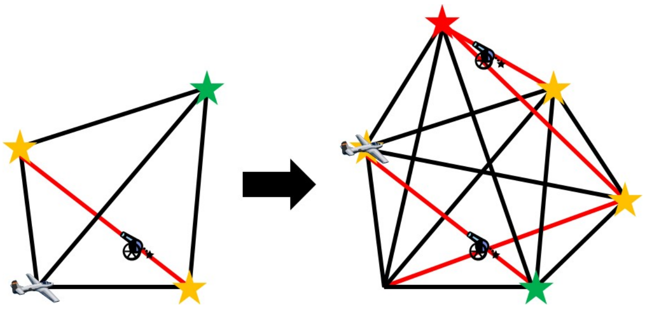

Figure 4.1: Example of UAV patrolling mission in an UDE.

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, and 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. 4.1. Thus, the UAV has to modify its initial plan to accomplish the mission. We aim, in this chapter, to develop a mission planner that can deal with the dynamic nature of such a problem.

4.2.2 Waypoints Finite State Machine

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. 4.2.

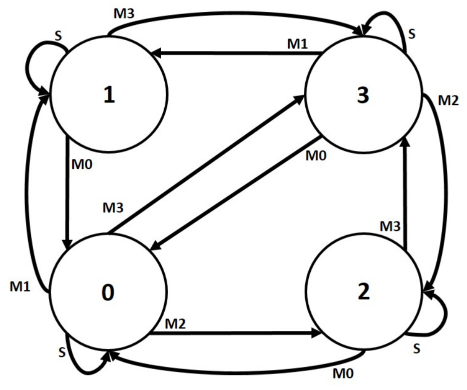

Figure 4.2: Waypoints FSM for a case with a single UAV and three waypoints.

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. 4.2 can be represented as a four-tuple:

$$
\begin{aligned}
X_W =& \{0, 1, 2, 3\}, \\
U_W =& \{S, M_0, M_1, M_2, M_3\}, \\
x_{W_0} =& 0, \\
x_W(k+1) =& f_W(x_W(k), u_W(k)),
\end{aligned} \tag{4.1}
$$

where f_W operates as shown in Table 4.1.

Table 4.1: 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

4.2.3 Task Finite State Machine

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. 4.3.

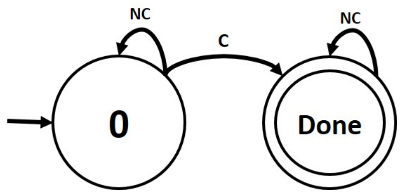

Figure 4.3: Task FSM for single waypoint.

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 ‘not checked waypoint’. The initial state is 0 for all waypoints. The task FSM can be represented as a four-tuple:

$$
\begin{aligned}
X_T =& \{0, Done\}, \\
U_T =& \{C, NC\}, \\
x_{T_0} =& 0, \\
x_T(k+1) =& f_T(x_T(k), u_T(k)),
\end{aligned}
\tag{4.2}
$$

where f_T operates as shown in Table 4.2.

Table 4.2: Task FSM state transition table.

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

4.2.4 Mission Restricted Finite State Machine

In order to form an 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 an 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.

4.2.4.1 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 waypoints FSM with three task FSMs (in our 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:

$$\begin{aligned}
X_U =& X_W \times X_{T_1} \times X_{T_2} \times X_{T_3}, \\
U_U =& \{u_U \in U_W \times U_{T_1} \times U_{T_2} \times U_{T_3} : \mathit{action}(u_U) \leq 1\}, \\
x_{U_0} =& x_{W_0} \times x_{T_{1_0}} \times x_{T_{2_0}} \times x_{T_{3_0}}, \\
f_U =& (f_W, f_{T_1}, f_{T_2}, f_{T_3}),
\end{aligned} \tag{4.3}$$

where

$$\begin{aligned}
& f_U(x_U, u_U) = \emptyset, \text{ when} \\
& u_U = (\bullet, C, \bullet, \bullet) \in U_U, x_U \neq (1, \bullet, \bullet, \bullet) \in X_U, \text{ and} \\
& u_U = (\bullet, \bullet, C, \bullet) \in U_U, x_U \neq (2, \bullet, \bullet, \bullet) \in X_U, \text{ and} \\
& u_U = (\bullet, \bullet, \bullet, C) \in U_U, x_U \neq (3, \bullet, \bullet, \bullet) \in X_U,
\end{aligned} \tag{4.4}$$

where $\mathit{action}(u_U)$ 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 (4.4) means 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 (4.3) 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.

4.2.4.2 Multi UAV

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

$$\begin{aligned}
X_{MU} =& X_{U_1} \times X_{U_2} \times \cdots \times X_{U_n}, \\
U_{MU} =& \{u_{MU} \in U_{U_1} \times U_{U_2} \times \cdots \times U_{U_n} : u_{U_1} \neq u_{U_2} \neq \cdots u_{U_n}\}, \\
x_{MU_0} =& x_{U_{1_0}} \times x_{U_{2_0}} \times \cdots \times x_{U_{n_0}}, \\
f_{MU} =& (f_{U_1}, f_{U_2}, \cdots, f_{U_n}).
\end{aligned} \tag{4.5}$$

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 (4.5): 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 is for the waypoints FSM for the first UAV, the next three inputs are for each task FSM for the first UAV, the fifth input is for the waypoints FSM for the second UAV, and the other three inputs are for each task FSM for the second UAV. By doing this, each UAV can have its individual input 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.

4.3 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

$[0, X_{area}] \times [0, Y_{area}]$ 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:

$$\begin{aligned} \mathcal{PP}(x_{T_i}, u_{T_i}) &= R, \text{ when } x_T = Done, \\ \mathcal{PP}(x_{T_i}, u_{T_i}) &= p_i \times P, \text{ when } x_T = 0, \end{aligned} \tag{4.6}$$

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:

$$\mathcal{W}(x_U, u_U) = c(x_W, u_W) + \sum_{i=1}^{m} \mathcal{PP}(x_{T_i}, u_{T_i}). \tag{4.7}$$

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

$$\mathcal{W}(x_{MU}, u_{MU}) = \mathcal{W}(x_{U_1}, u_{U_1}) + \cdots + \mathcal{W}(x_{U_n}, u_{U_n}). \tag{4.8}$$

For accessible analysis, we assign negative values to task FSM transition rewards, R in (4.6). Thus we can define the UDE patrolling problem as minimizing ReRFSM transition weights.

Definition IV.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, $\mathcal{W}$.

4.4 Policy and Solution Approach

4.4.1 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 as shown in Sec. 2.4. However, for RFSMs, this does not hold due to the restricted inputs in RFSM as shown in Sec. 2.5. Thus, the optimal policy of RFSMs has to be obtained by applying DP directly to the whole RFSM. Defining FSM in the state space as:

$$RFSM = (\mathcal{ST}, \mathcal{IP}, f, Init), \tag{4.9}$$

where $\mathcal{ST}$ and $\mathcal{IP}$ are sets representing X_{MU} and U_{MU} in (4.5) respectively, let the function f: $\mathcal{ST} \times \mathcal{IP} \rightarrow \mathcal{ST}$ be the next state function, and $Init \in \mathcal{ST}$ 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:

$$st(k+1) = f(st(k), \mathcal{T}_D(k)), \tag{4.10}$$

and $st(0)$ is $Init$. Given the state transition mapping (4.10), consider the objective function:

$$J(st(0), \cdots, \mathcal{T}_D(0), \cdots) = \sum_{k=0}^{K-1} \mathcal{W}(st(k), \mathcal{T}_D(k)), \tag{4.11}$$

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:

$$J^*(st(k)) = \min_{\mathcal{T}_D(k) \in \mathcal{IP}} \{\mathcal{W}(st(k), \mathcal{T}_D(k)) + J^*(f(st(k), \mathcal{T}_D(k)))\}, \tag{4.12}$$

where J^* is the optimal cost-to-go from state st. From (4.10) and (4.12),

$$J^*(st(k)) = \min_{\mathcal{T}_D(k) \in \mathcal{IP}} \{\mathcal{W}(st(k), \mathcal{T}_D(k)) + J^*(st(k+1))\}. \tag{4.13}$$

Equation (4.13) illustrates that DP proceeds backwards with respect to steps. The optimal decision is then

$$\mathcal{T}_D^*(st(k)) = \underset{\mathcal{T}_D(k) \in \mathcal{IP}}{\text{argmin}} \{\mathcal{W}(st(k), \mathcal{T}_D(k)) + J^*(st(k+1))\}. \tag{4.14}$$

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^*) [104, 10]. Then, the optimal policy for FSM is:

$$\pi^* = DP(RFSM, \mathcal{W}, K), \tag{4.15}$$

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 $st \in \mathcal{ST}$.

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 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 for the patrolling mission is shown in Algorithm 5.

Algorithm 5 Algorithm of DP for ReRFSM

```
 1: $\pi^*(k) \leftarrow$ DP for ReRFSM$(w(k), w(k-1), L(k), L(k-1))$
 2: **if** $w(k) \neq w(k-1)$ **or** $L(k) \neq L(k-1)$ **then**
 3:    $m \leftarrow$ length$(w(k))$
 4:    **for** $i = 0$ **to** $m$ **do**
 5:       **if** $i = 0$ **then**
 6:          $FSM_i \leftarrow$ waypoints FSM
 7:       **else**
 8:          $FSM_i \leftarrow$ task FSM of $w_i(k)$
 9:       **end if**
10:    **end for**
11:    $RFSM \leftarrow FSM_0 \bar{\times} FSM_1 \bar{\times} ... \bar{\times} FSM_m$
12:    $\mathcal{W} \leftarrow$ transition weight of RFSM
13:    **for** $i = 1$ **to** $m$ **do**
14:       **if** $FSM_i$ needs to be pruned **then**
15:          $RFSM \leftarrow RFSM \,\bar{/}\, FSM_i$
16:          $\mathcal{W} \leftarrow$ transition weight of RFSM
17:       **end if**
18:    **end for**
19:    $\pi^*(k) =$ DP$(RFSM, \mathcal{W}, K)$
20: **else**
21:    $\pi^*(k) = \pi(k-1)$
22: **end if**
23: **return** $\pi^*(k)$
```

4.4.2 LBFS for ReRFSM

The motivation of this section is that DP suffers from the curse of dimensionality. Using DP for ReRFSM is computationally 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 as described in Sec. 2.7. In such a situation, obtaining a policy is useless, and obtaining a solution may be better. Thus, we use BFS or LBFS algorithm with fixed depth (fd) to get the solution for the current state as we described in Sec. 2.6.2. The algorithms for LBFS for ReRFSM for the patrolling mission is shown in Algorithm 6.

Algorithm 6 Algorithm of LBFS for ReRFSM.

```
$u(k) \leftarrow$ LBFS for ReRFSM($w(k), L(k),$)
$m \leftarrow$ length($w(k)$)
for $i = 0$ to $m$ do
    if $i = 0$ then
        $FSM_i \leftarrow$ waypoints FSM
    else
        $FSM_i \leftarrow$ task FSM of $w_i(k)$
    end if
end for
$RFSM \leftarrow FSM_0 \;\bar{\times}\; FSM_1 \;\bar{\times}\; ... \;\bar{\times}\; FSM_m$
$\mathcal{W} \leftarrow$ transition weight of RFSM
for $i = 1$ to $m$ do
    if $FSM_i$ needs to be pruned then
        $RFSM \leftarrow RFSM \;\bar{/}\; FSM_i$
        $\mathcal{W} \leftarrow$ transition weight of RFSM
    end if
end for
$u(k)$ = LBFS($RFSM$, $\mathcal{W}$, $fd$, $CC_{ub}$)
return $u(k)$
```

4.5 Analysis of the Approach

4.5.1 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.

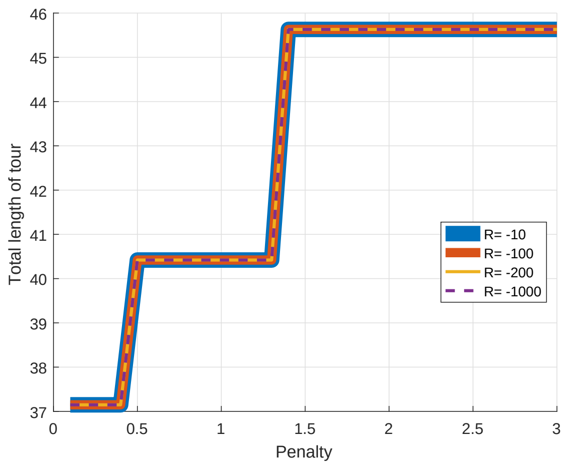

Figure 4.4: Total length of tour for different penalty (P) and reward (R).

P is 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.

4.5.2 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$^{\text{TM}}$ i5-4300U Processor CPU 1.90GHz, 8.0GB RAM.

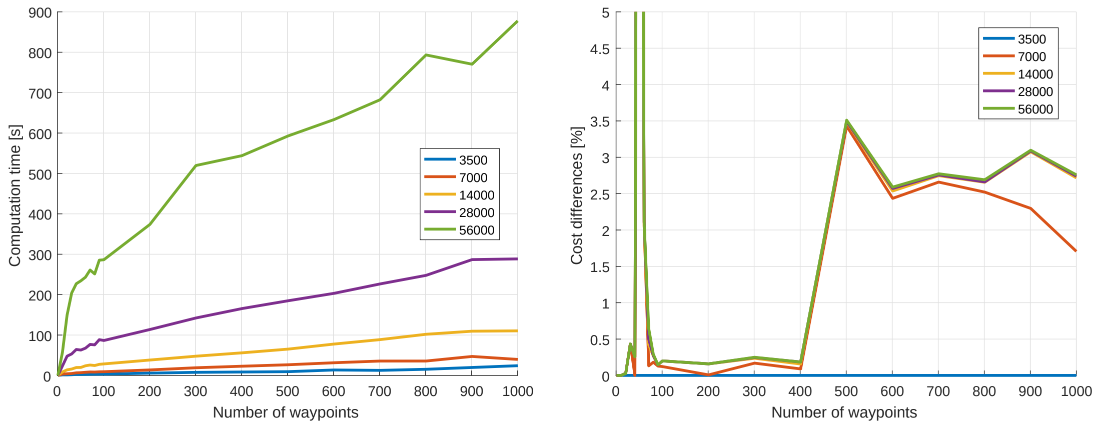

Figure 4.5: Comparison of computation times and costs.

For the comparison, we choose $fd = 10$. The cost comparison is shown in the lower figure in Fig. 4.5. 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 $CC_{ub} = 3,500$ in percentage, so it is zero for $CC_{ub} = 3,500$. The cost difference between the case of $CC_{ub} = 3,500$ and the others are less then 0.5% for 400 waypoints and less than 4% for up to 1,000 waypoints. However, the computation time, as shown in the upper figure in Fig. 4.5, for the case of $CC_{ub} = 3,500$ 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, $CC_{ub} = 3,500$ 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.

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 $CC_{ub} = 3,500$ for LBFS. The computation time comparison is shown in Fig. 4.6.

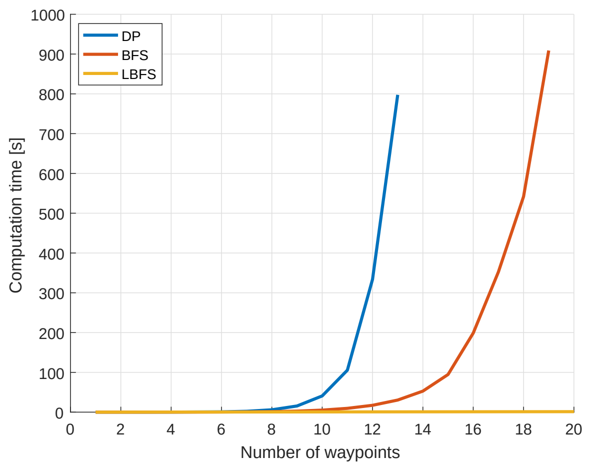

Figure 4.6: Comparison of computation time of DP, BFS and LBFS.

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.

4.6 Simulations and Results

In this section, every simulation is performed in the environment described in the previous section. For the simulation, 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 $CC_{ub} = 3,500$.

4.6.1 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. 4.7. The length of the tour is 14.45, which is the exact minimum length of the tour for the given waypoints.

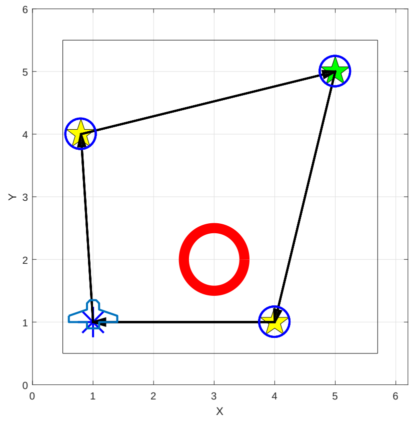

Figure 4.7: Initial plan.

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. 4.8. Complying with the new no fly zone, priorities, and waypoints added, the total length of the tour is 23.53.

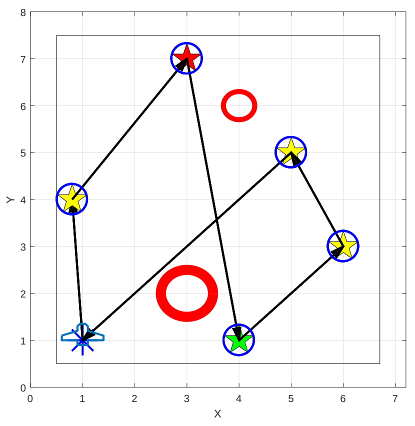

Figure 4.8: Updated plan.

4.6.1.1 Initial plan with different settings

The initial plans, without priorities, and without an unavailable path, are shown in Fig. 4.9. When there is no priorities as shown in the left subfigure of Fig. 4.9, the tour is similar to the 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. 4.9, 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.

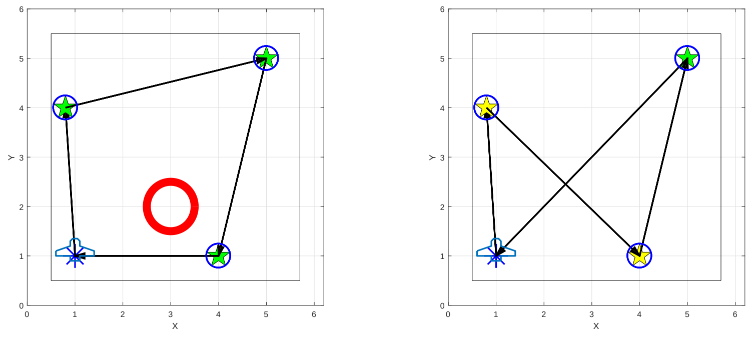
Figure 4.9: Initial plans with different settings: without priorities or no fly zones.

4.6.1.2 Updated plan with different settings

The updated plan, without priorities and without unavailable paths, is shown in Fig. 4.10. When there is no priority as shown in the left subfigure of Fig. 4.10, 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. 4.8. When all paths are available, as shown in the right subfigure of Fig. 4.10, 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.

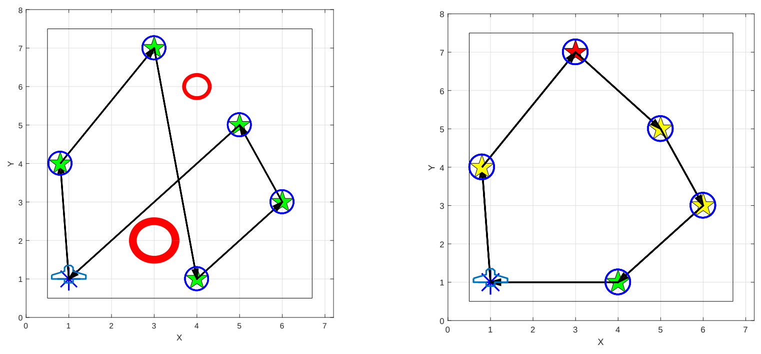
Figure 4.10: Updated plans with different settings: without priorities or no fly zones.

4.6.2 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 contains eight waypoints with different priorities where the locations and the priorities of the waypoints are randomly decided. We do not consider no fly zones in this section. The simulation results are shown in Table 4.3.

Table 4.3: Comparison of DP and LBFS.

	DP	**LBFS**
Average tour length	18.2439	18.2439
Average computation time	15.21 [s]	0.36 [s]
Number of computations	once	every step

Because there are no environment changes and DP generates the optimal policy, only one computation occurs for DP, but LBFS generates the solution for every time step. However, the average computation time for LBFS is very fast, so it can compute the solution at every time step. 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, and both generate the same plans. This indicates that for small scale problems, $fd = 10$ and $CC_{ub} = 3,500$ for LBFS generates the same solution with DP.

4.6.3 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

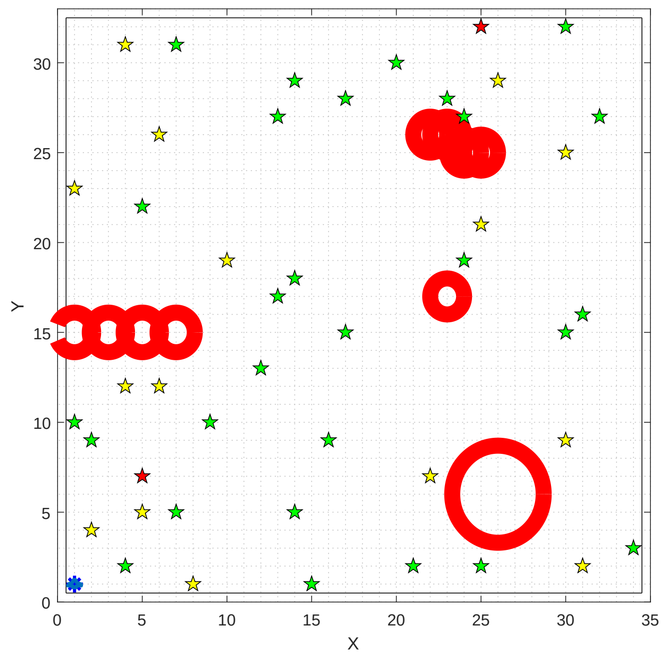

Figure 4.11: Large scale problem example.

likely generates sub-optimal solutions. We consider a scenario with 45 waypoints with different priorities, and no fly zones as shown in Fig. 4.11.

For this simulation, we choose $fd = 20$. The LBFS for ReRFSM successfully generates a plan, where the length of the tour is 350.39. The average computation time is 9.8957 seconds, which is acceptable. 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 232.75 and the length of the second case is 275.26. 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 priorities 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.

4.6.4 Two UAVs: Small Scale Example

We simulate a case with two UAVs and four waypoints as shown in Fig. 4.12 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.

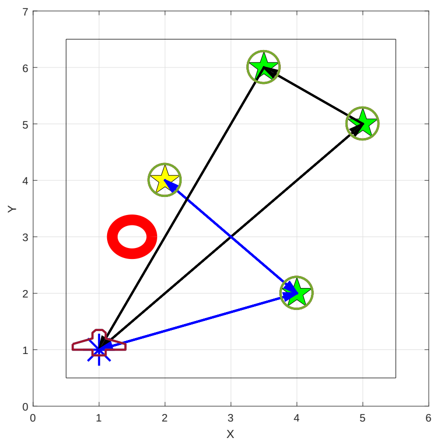

Figure 4.12: Plan for two UAVs with four waypoints.

4.7 Conclusions

A mission planner for an 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 approximately 400 waypoints in the current setting, 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.

CHAPTER V

Coordinated Model Predictive Control of Aircraft Gas Turbine Engine and Electrical Power System

5.1 Introduction

5.1.1 Motivation

In the past few decades, the electrical power requirements for aircraft have been steadily increasing, concomitant with trends towards MEA and AEA [71]. A typical aircraft power system involves one or more generators connected to one or more gas turbine engines, integrated with energy storage elements that provide supplemental electrical power, a distribution system and loads. The large electrical loads, including both steady and transient loads, affect the operation of the generators and of the gas turbine engines. For instance, large electrical load changes induce large torque disturbances on the gas turbine engine and can affect engine thrust and shaft speeds. These changes, in turn, affect the generators; hence, the system exhibits strong static and dynamic interactions. Thus, in the presence of large electrical loads, the interactions between the electrical system and gas turbine engine have to be addressed for efficient and safe operation of aircraft.

Our objective is to establish an integrated, model-based control capability for an aircraft's propulsion and electrical power systems, including thrust generation, electric power generation, and energy storage, that improves the capability of the system to accommodate large transients, including those caused by large transient electrical loads, while maintaining the operation of the components and the overall system within a specified safe range by enforcing appropriately defined state and control constraints.

Specifically, in this chapter, the development of an integrated control system is considered that accommodates large steady and transient electrical loads, maintains aircraft flight performance by delivering requested thrust, enforces gas turbine engine constraints (e.g., surge margins), as well as electrical system constraints (e.g., component power limits), and reduces fuel consumption. To facilitate the achievement of these goals, an advanced two-shaft distributed generator configuration is considered where one generator is connected to the High Pressure Shaft (HPS) and the other is connected to the Low Pressure Shaft (LPS) of the gas turbine engine. This configuration affords an extra degree of freedom to accommodate the effects of large loads compared to the single-shaft configuration. Furthermore, it potentially achieves better fuel efficiency than the single-shaft configuration. In addition, the integration of high performance storage elements that can react quickly to transient loads to assist the generators and the gas turbine engines is considered.

To control such an advanced system with two generators, a gas turbine engine and energy storage, we define a power split strategy between the two generators based on the offline minimization of the fuel consumption, and a rule-based strategy to determine when to charge and discharge the energy storage. To protect the engine and the electrical system components against constraint violation, a rate-based MPC framework is exploited and several MPC controller designs are developed, validated on a nonlinear model of the system and compared with each other. The proposed framework is flexible and modular and can accommodate other constraints not explicitly treated in the chapter, such as temperature constraints in the engine or voltage stability constraints in the electrical systems, provided the prediction model is updated with representations for these constraints. Since only linear MPC design techniques are employed, the controller implementation is feasible with standard quadratic programming solvers that are becoming a mature and reliable technology.

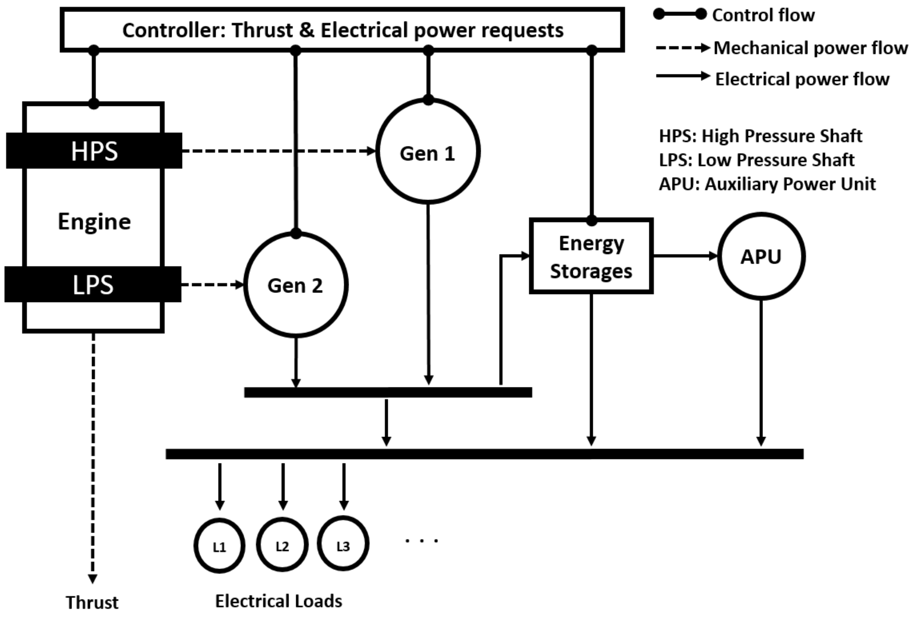

Figure 5.1: A schematic of the gas turbine engine and the electrical power system.

The system configuration of interest in this chapter is illustrated in Fig. 5.1. The system consists of a single gas turbine engine, energy storage element(s), and two generators, one of which is attached to the LPS of the gas turbine engine while the other is attached to the HPS of the gas turbine engine.

5.1.2 Literature Review

The growing electrical power requirements of MEA and AEAs are highlighted in [71, 83]. For instance, at least 1.6 MW will be required for a next-generation 300 pax aircraft [83]. Large electrical power is required for turboelectric propulsion. Three MW generators are considered in [59] and a 40.2 MW generator is planned in [39, 51]. Electrical weapons systems for military applications also require large electrical power, from 0.025 to 4.5 MW depending on the type [66]. Directed energy systems are one of the key 12 potential capability areas for the U.S. Air Force [86]. To deal with these large electrical loads on aircraft, integrated control of the aircraft's gas turbine engine, electrical power system, and thermal management are necessary. Challenges in aircraft engine control and integrated power and thermal management are discussed in [6, 7, 58, 27].

Since our system has constraints, we employ MPC [81]. MPC-based approaches have been considered to develop solutions to many recent control problems, including gas turbine engine control, see e.g. [82, 25, 42, 3]. Rate-based MPC allows setpoint tracking and has been applied to turbofan engine clearance control in [25] and to turbocharged compression ignition engine control in [47]. References [53, 54] report the application of reference and extended command governors to handle constraints in gas turbine engines. In this chapter, the Multi-Parametric Toolbox (MPT) [55] is employed for computational implementation of a rate-based MPC controller.

The two-generator configuration for aircraft, with one generator connected to the HPS and the other generator connected to the LPS of a gas turbine engine, is introduced in [70]. The challenges and possible research directions for UAVs and MEA with a gas turbine engine, two-generator configuration, and battery (and/or supercapacitor) are discussed in [79]. The authors of [79] indicate the necessity of integrated control of the electrical system and the gas turbine engine system due to interactions between both systems. In [74], the authors design a voltage and current controller for the generators and this work is extended in [4] to include a battery. The controller proposed in these references is based on a master-slave configuration for high-load situations. Existing publications on two-generator configurations focus primarily on the electrical system, especially voltage and current stability, and a control design exploiting batteries.

Integrated control of a gas turbine engine and electrical power system has been considered in some publications. The Nonlinear Model Predictive Control (NMPC) approach for a 166 MW heavy-duty single shaft gas turbine power plant based on simplified gas turbine engine and generator models is presented in [52]. The control goal is to supply all electrical loads, maintain the rotor speed, exhaust gas temperature, and turbine firing temperature by controlling air flow and fuel flow despite transient load changes. The control of the gas turbine engine and electrical system, focused on their thermal management, for the U.S. Navy's future all-electric ship is considered in [94]. The importance of interactions between the gas turbine engine and the electrical system for aircraft are highlighted in [75] where the engine response when a step change reduction of electrical power occurs is simulated. In [97], an energy storage element (supercapacitor) is used to reduce the effects of high dynamic loads on the engine using a Proportional-Integral (PI) supercapacitor controller. A load management system, which consists of generators, contactors, buses and loads, and a battery for an aircraft electric power system, is presented in [90]. The paper [90] focuses on the electrical system of the aircraft, mainly controlling contactors for safety and reliability, using load shedding. The aircraft gas turbine engine modeling and control are discussed in [48].

A Simulink-based Toolbox for the Modeling and Analysis of Thermodynamic Systems (T-MATS) [18, 107, 17] is used for the gas turbine engine modeling and is supplemented by an electrical power system model in Simulink. T-MATS allows one to model both steady state and dynamic gas turbine engine operation.

5.1.3 Original Contributions

The original contributions of this chapter are as follows:

1. We treat an advanced electrical power system configuration with two generators and an electrical storage element, and we successfully design and demonstrate control designs which accomplish simultaneous tracking of requested thrust com-

mands and electrical power output commands, while satisfying the imposed component protection constraints within the engine and the electrical system, and minimizing the fuel consumption. These designs account for static and dynamic interactions between the gas turbine engine, generators and energy storage. We also illuminate the link between energy storage characteristics and control performance.

2. We define a novel control system architecture based on a combination of a rate based MPC controller, a power split map between generators optimized for steady-state operation and a supervisory logic to govern energy storage charging/discharging. The benefits of constrained coordinated control include the ability to handle load pulses of higher frequency and larger magnitude than possible with existing systems.

3. We also propose a novel linear transformation approach to match states of different linear prediction models from system identification. This approach avoids the need for designing observers for non-physical states of the individual models.

4. We compare MPC controller designs based on single linear and multiple linear prediction models where the linear prediction models are determined by applying system identification techniques. We demonstrate that the mismatch between the linear prediction models and the actual nonlinear system can be successfully handled by auxiliary offset states so that the surge margin constraints can be robustly enforced.

5. We demonstrate that successful control of the engine can be accomplished utilizing a single rate-based linear prediction model with auxiliary offset states that allows lower computational and implementation complexity. We compare the controller performance with the energy storage and verify the benefits of

the energy storage to the system. We validate the design in nonlinear model simulations over the full engine operating range, while responding to large transient thrust commands and electrical power loads. The proposed control design framework is systematic and expandable, e.g., to incorporate additional constraints or components.

5.2 Modeling

In this section, models of the gas turbine engine, generators and energy storage elements are described. A simple relationship between the shaft speeds of the gas turbine engine and the output power of the generators is used, assuming the dynamics of the generators are much faster than the dynamics of the gas turbine engine, and a first-order model is adopted to represent the dynamics of the energy storage elements. The engine model, generator models, and energy storage element models are assembled into a system level model in which one generator is connected to the HPS and the other generator is connected to the LPS of the gas turbine engine. Note that the assembled model is able to represent subsystem level interactions visible in the simulation results.

Then, a linear model of the gas turbine engine with two generators is obtained via system identification followed by a linear transformation of all the states of the linear model to physical states. Note that the identified linear model takes into account the interactions between the gas turbine engine and the generators. Finally, the identified linear model, the generator models, and energy storage element models are combined to obtain the complete linear prediction model to be used in MPC control design.

5.2.1 Gas Turbine Engine

The JT9D gas turbine engine model provided with T-MATS package [107] is used to represent engine dynamics. T-MATS is a Simulink-based tool for thermodynamic system simulation developed and released by NASA to facilitate research involving gas turbine engine simulations and control of the kind pursued in this chapter. Unlike other packages, T-MATS is open to public use. It includes generic modeling libraries and is suitable for gas turbine engine modeling. The JT9D gas turbine engine model represents the dynamics of shaft speeds, pressures and flows in various components of the engine and predicts engine thrust. The model is developed and verified based on data from the Numerical Propulsion System Simulation (NPSS) [17]. The thrust (Fg) is controlled using the Fuel to Air Ratio (FAR) as a control input.

5.2.2 Generators

The two generators are each connected to different shafts of the gas turbine engine: one to the HPS and one to the LPS. We refer to the generator that is connected to the HPS as the High Pressure Shaft Generator (HPSG) and the generator that is connected to the LPS as the Low Pressure Shaft Generator (LPSG). Then, the power requested from the HPSG (P_{Hreq}) and the power requested from the LPSG (P_{Lreq}) are two additional control inputs in our system. The total output power from the generators (P_{G_T}) is the sum of the output powers of the HPSG (P_H) and LPSG (P_L). The power difference between two generators (P_D) is one of the outputs of the system and is defined as $P_H - P_L$.

Assuming that the dynamics of the generators are much faster than those of the gas turbine engine [21], a simple relationship between the shaft speeds of the gas turbine engine and the output power of the generators is adopted based on given efficiencies of the generators,

$$\begin{aligned} P_H &= N_H \times \tau_{E_H} \times \eta_H, \\ P_L &= N_L \times \tau_{E_L} \times \eta_L, \end{aligned} \tag{5.1}$$

where N_H, τ_{E_H} and η_H are, respectively, the shaft speed, torque on the shaft and efficiency of the HPSG, and N_L, τ_{E_L} and η_L are, respectively, the shaft speed, torque on the shaft and efficiency of the LPSG. Thus, given electrical power outputs of the generators, the torques that the generators create on the gas turbine engine shafts can be computed according to

$$\begin{aligned} \tau_{E_H} &= \tfrac{P_H}{N_H \times \eta_H}, \\ \tau_{E_L} &= \tfrac{P_L}{N_L \times \eta_L}. \end{aligned} \tag{5.2}$$

Note that the above electrical power system representation is suitable given the specific control objectives in this chapter and is justified by the time-scale separation between the engine dynamics and the dynamics in the electrical power system. In the subsequent analysis and simulations, constant values of the efficiencies, $\eta_H = \eta_L = 0.9$, are assumed.

5.2.3 Energy Storage Elements

The energy storage element model is as follows:

$$\frac{dE_j}{dt} = -P_j, \tag{5.3}$$

where E_j is the total energy stored in the energy storage j, P_j is power to/from the energy storage j, and j indicates the type of energy storage element. In this chapter, a battery and/or ultracapacitor are exploited as the energy storage elements, so $j \in \{B, C\}$ where B indicates the battery and C indicates the ultracapacitor. Then, the State of Charge (SoC) is given by

$$SoC_j = \frac{E_j}{E_{j_{Max}}}, \tag{5.4}$$

where $E_{j_{Max}}$ is the maximum energy that can be stored in the energy storage j. The total output/input power of the energy storage elements (P_{ES_T}) is the sum of the output/input power of all the energy storage elements. Then, the total output power, P_T, is the sum of the total output powers from the generators (P_{G_T}) and the total output/input powers of the energy storage elements (P_{ES_T}).

5.2.4 Linear Design Model

5.2.4.1 System Identification and Linear Transformation

The design of our MPC controller is based on a linear prediction model. Since our gas turbine engine model is essentially of black-box type, either analytical or numerical (finite-difference-based) linearization cannot be easily implemented. Consequently, the linear model is identified based on the input-output response data collected from the nonlinear model of the engine near a nominal operating point. The nominal operating point is the same as the one used for verifying the model in [17] ($27,593$ [lbf] thrust and FAR of 0.0187) and $P_{Hreq} = P_{Lreq} = 0$ [MW].

Our linear model to be identified has three inputs, FAR, P_{Hreq}, and P_{Lreq}, and five outputs, the HPS speed, LPS speed, thrust, Low Pressure Compressor (LPC) surge margin, and High Pressure Compressor (HPC) surge margin. The surge margins are added as outputs to the model to predict the evolution of the surge margin constraints over the prediction horizon.

To identify the linear prediction model at a given operating point, a system identification approach is followed. The input-output data set is based on a 400 sec trace generated when chirp signals are applied to each of FAR, P_{Hreq}, and P_{Lreq} channels for 100 sec individually, and then to all inputs in combination for another 100 sec. The magnitude of chirp signals is set to 0.001 for δFAR and 0.5 for δP_{Hreq} and δP_{Lreq} where δ designates the deviation from steady-state values at the operating point. The chirp signal frequency ranges between 0 Hz and 1.8 Hz. After the set of input-output data is obtained by simulating the nonlinear model, mean removal is applied so that only variations from the steady state are reflected in the signals.

Based on such input-output data collected around a specific operating point, the linear model of order five is identified using the system identification toolbox in Matlab, and is verified to be both asymptotically stable and fully controllable. This identified linear model has the following form,

$$\begin{aligned} \delta\dot{x} &= A\delta x + B\delta u, \\ \delta y &= C\delta x, \end{aligned} \tag{5.5}$$

where δx is the state, δu is the input deviations from the operating point, δy is the output deviations from the operating point, $A \in \mathbb{R}^{5\times 5}$, $B \in \mathbb{R}^{5\times 3}$, and $C \in \mathbb{R}^{5\times 5}$. The resulting linear model from system identification typically has $C \neq I$, which indicates that the states are not physical. Since models with physical states have advantages in terms of state estimation (e.g., non-physical states must be estimated even if physical states are measured) and control design (e.g., switching between different linear state feedback controllers is straightforward), a state transformation is constructed to obtain $C = I$. Specifically, let $\delta z = \delta y$, so δz is the physical state. Then,

$$\delta z = C\delta x \;\Rightarrow\; C^{-1}\delta z = \delta x \;\Rightarrow\; \dot{C}^{-1}\delta z + C^{-1}\delta\dot{z} = \delta\dot{x}. \tag{5.6}$$

Substituting for $\delta\dot{x}$ from Eq. (5.5) yields

$$\dot{C}^{-1}\delta z + C^{-1}\delta\dot{z} = A\delta x + B\delta u \;\Rightarrow\; C^{-1}\delta\dot{z} = A\delta x + B\delta u. \tag{5.7}$$

Since $\delta x = C^{-1}\delta z$,

$$C^{-1}\delta\dot{z} = AC^{-1}\delta z + B\delta u \;\Rightarrow\; \delta\dot{z} = CAC^{-1}\delta z + CB\delta u. \tag{5.8}$$

Let $A' = CAC^{-1}$, $B' = CB$, and $\delta x = \delta z$. Then, the new system is as follows:

$$\begin{aligned} \delta\dot{x} &= A'\delta x + B'\delta u, \\ \delta y &= C'\delta x = I\delta x, \end{aligned} \tag{5.9}$$

where now δx is the physical state, and

$$\delta x = \begin{bmatrix} \delta x_{N_H} \\ \delta x_{N_L} \\ \delta x_{Fg} \\ \delta x_{SM_{LPC}} \\ \delta x_{SM_{HPC}} \end{bmatrix}, \quad \delta u = \begin{bmatrix} \delta FAR \\ \delta P_{Hreq} \\ \delta P_{Lreq} \end{bmatrix}. \tag{5.10}$$

Here δx_{N_H} is the HPS speed deviation, δx_{N_L} is the LPS speed deviation, δx_{Fg} is the thrust deviation, $\delta x_{SM_{LPC}}$ is the LPC surge margin deviation, and $\delta x_{SM_{HPC}}$ is the HPC surge margin deviation. The components of the control input vector are δFAR, δP_{Hreq}, and δP_{Lreq} and they represent the deviations in the respective inputs. Note that choosing the order of the linear model equal to five is essential for this transformation procedure to apply.

To confirm linear model accuracy, we have generated another 100 sec trace of input-output data for validation purposes. This trace was constructed similarly to the one used to generate system identification data but with the chirp signals frequency range being between 0 Hz and 3.2 Hz, and chirp signals were applied to all inputs channels in combination for 100 sec. The agreement between the validation data and the identified linear model is 81.34% for HPS speed, 80.24% for LPS speed, 81.41% for thrust, 63.80% for LPC surge margin, and 82.62% for HPC surge margin. The agreement is defined in terms of normalized root mean square error as

$$\text{agreement [\%]} = 100 \times \left(1 - \frac{||y - \hat{y}||}{||y - y_{avg}||}\right), \tag{5.11}$$

where y is the measurement vector, $\hat{y}$ is the estimate vector y_{avg} is the mean of y, and $||\cdot||$ denotes the 2-norm applied to the respective vectors of measurements/estimates.

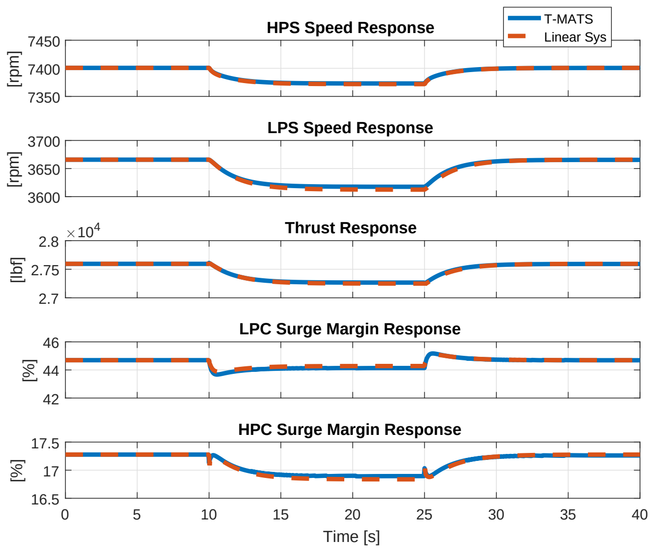

Figure 5.2: Comparison of step responses of the linear and nonlinear models.

Fig. 5.2 compares step responses of the linear model and T-MATS. These results were obtained at the operating point corresponding to $FAR = 0.0187$, and $P_{Hreq} = P_{Lreq} = 0$. The T-MATS initially runs at the steady state, then step increments of the inputs, $\delta FAR = 0.0001$, $\delta P_{Hreq} = 0.1MW$, and $\delta P_{Lreq} = 0.1MW$, are applied during the time period between 10 and 25 sec. The agreement between the nonlinear model (T-MATS) and the identified linear model is 94.75% for HPS speed, 86.74% for LPS speed, 93.37% for thrust, 74.15% for LPC surge margin, 80.81% for HPC surge margin, and the average is 85.96%. Note that if the response of surge margins is not considered, the average agreement for the step responses between the nonlinear model and the identified linear model increases to 91.62%, which is fairly accurate. A comparably larger mismatch of the surge margin response prediction is compensated by the auxiliary offset states (see Sec. 5.4.4.2 and Sec. ??). Furthermore, our controller is feedback-based and feedback compensates for model inaccuracies.

To confirm model accuracy, we checked the sensitivity of the results to the choice of signals used for identification. Specifically, we considered 19 other random frequency sub-ranges (within the overall 0 Hz to 2.4 Hz range) for the chirp signal which was used to generate input-output data for identification. This did not substantially change the results against the validation data.

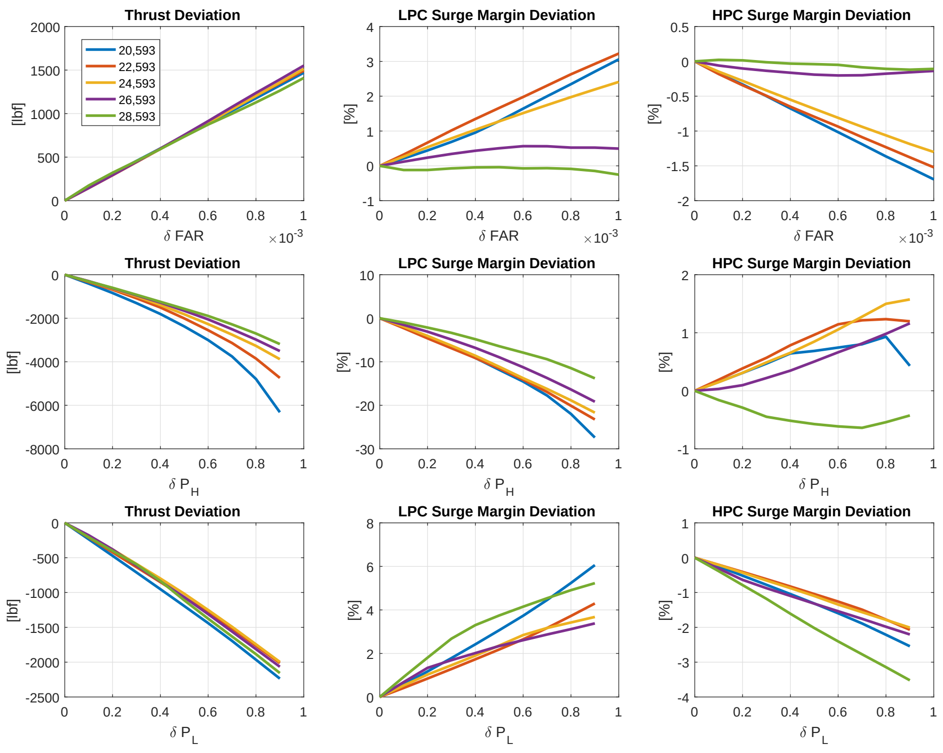

Figure 5.3: Steady states values of the nonlinear model for different thrust levels.

Steady states values of thrust, LPC surge margin and HPC surge margin deviations as functions of different δFAR, δP_H, and δP_L for different operating points based on the nonlinear model are shown in Fig. 5.3. The different colors indicate different operating points (defined by different thrust levels) as indicated in the legends of the figures. As observed, the gas turbine engine with two generators is a highly nonlinear system. In particular, the static (dc) gains are different at different operating points defined by different thrust levels. Thus, multiple linear models may be needed to represent the response at different operating points.

5.2.4.2 Combined Linear Model

The linear model, Eq. (5.9), is combined with the generator and energy storage elements models. The outputs of the integrated system are the thrust (Fg), total power (P_T), power difference between the two generators (P_D) and stored energy in energy storage elements (E_j). The total power is $P_H + P_L + P_j = P_{Hreq} + P_{Lreq} + P_{jreq}$ and the power difference between the two generators is $P_D = P_H - P_L = P_{Hreq} - P_{Lreq}$. The combined model has the following form:

$$\begin{bmatrix} \delta\dot{x} \\ \dot{E}_j \end{bmatrix} = \begin{bmatrix} A' & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta x \\ E_j \end{bmatrix} + \begin{bmatrix} B' & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \delta u \\ P_{jreq} \end{bmatrix},$$

$$\begin{bmatrix} \delta F_g \\ \delta P_T \\ \delta P_D \\ \delta P_D \\ E_j \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x \\ E_j \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta u \\ P_{jreq} \end{bmatrix}. \tag{5.12}$$

For control purposes, two outputs for the power difference between two generators (P_D) are needed, as described in the next section. Thus, the inputs in Eq. (5.12) are δFAR, δP_{Hreq}, δP_{Lreq} and P_{jreq}, and the outputs in Eq. (5.12) are δF_g, δP_T, δP_D, δP_D and E_j.

5.3 Problem Formulation

In this chapter, the following problem formulation is considered: Given a gas turbine engine, energy storage elements, two generators, one connected to each shaft of the gas turbine engine, a requested thrust level and (large) expected/requested electrical loads, determine the fuel to air ratio of the gas turbine engine, input/output power of the energy storage elements and the electrical power output of each generator to supply all the required electrical loads, maintain the requested thrust level, and minimize fuel consumption, subject to surge margin limits and other constraints.

5.4 Controller Design

5.4.1 Overall Architecture

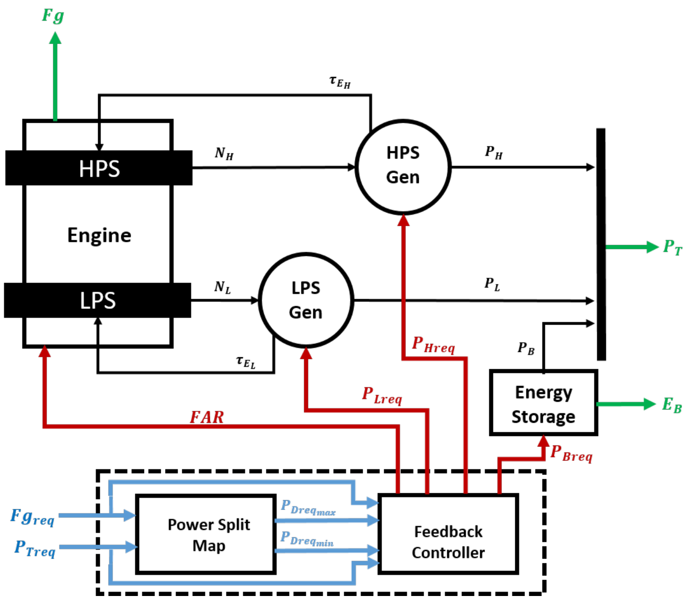

Figure 5.4: Control system architecture.

Our control architecture is shown in Fig. 5.4. The control system consists of a power split map and feedback controller designed as an MPC controller. The power split map determines the maximum and minimum optimal power differences ($P_{Dreq_{max}}$ and $P_{Dreq_{min}}$) between the two generators as a function of the requested thrust level (Fg_{req}) and a total electrical power (P_{Treq}) command. Then, the MPC controller generates the four control signals (FAR, P_{Hreq}, P_{Lreq}, P_{jreq}) to track the thrust, total electrical power and optimal power difference setpoints while enforcing system constraints.

5.4.2 Optimal Power Split

In this section, the gas turbine engine behavior and operating regions are analyzed for different electrical power loads and operating points in steady state based on the models described in Sec. 5.2. In particular, fuel consumption and compressor surge margins are considered.

In [87], the optimal power split map has been based on a point that minimizes fuel consumption for a given thrust and total electrical power output. In this chapter, we generalize this approach and define the optimal power split range in which the fuel consumption deviates from the optimal fuel consumption by no more than 0.3%. Examples of the fuel optimal power split ranges obtained by numerical optimization applied to our model for thrust levels of 21,593, 27,593 and 32,593 [lbf] are shown in Fig. 5.5.

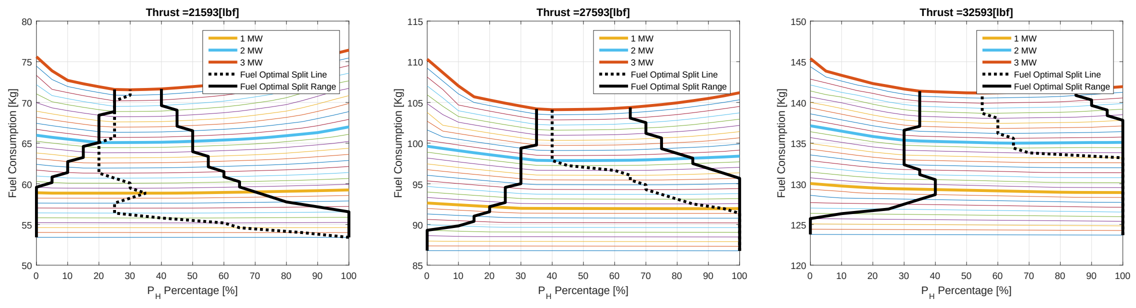

Figure 5.5: Fuel optimal power split range examples.

The colored lines correspond to different levels of total electrical power and they represent fuel consumption as a function of P_H percentage for a given total electric power level. The black dotted lines indicate the optimal P_H percentage where fuel consumption is minimal for given thrust and total electrical power output. The black solid lines indicate the interval of P_H percentage values within which the fuel consumption is not worse than 0.3% of optimal; this interval changes depending on the total electric power level and thrust. Thus, staying within the fuel optimal split range (between the black solid lines) yields good fuel efficiency, that is, no more than

0.3% worse than that for the fuel optimal split line (black dotted lines).

As observed, when the total electrical power is small, the fuel optimal power split range is large: we have much control flexibility. However, when the total electrical power is large, the fuel optimal power split range is small and hence an accurate control strategy is necessary for fuel efficient operation at large electrical power levels.

The safe operation of the gas turbine engine also has to be ensured. Thus, an additional requirement to maintain sufficient fan, LPC, and HPC surge margins is considered in the definition of the power split range. Specifically, 15% as the minimum surge margin for the fan, 20% as the minimum surge margin for LPC and 14% as the minimum surge margin for HPC are chosen for our control design and simulation-based case studies.

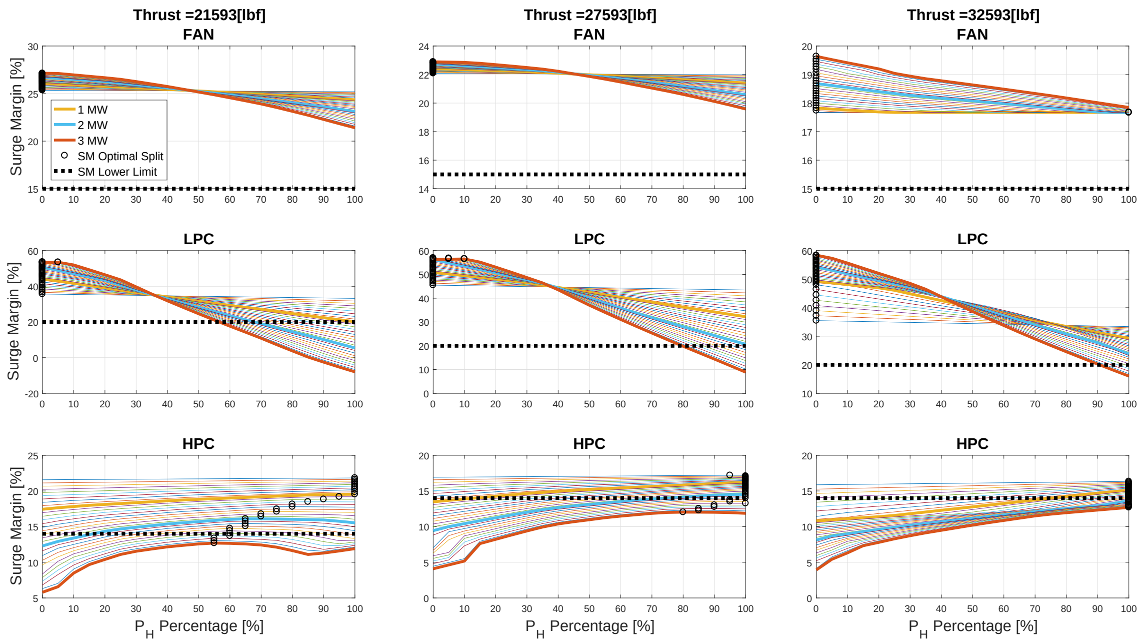

Figure 5.6: The surge margin dependence on other variables.

The surge margins as functions of P_H percentage at different levels of thrust and electrical power are shown in Fig. 5.6. The black circles indicate the power split that yields the highest surge margin for given thrust and electrical power output, and the black dotted lines indicate the surge margin lower bounds for each compressor. Thus, if the black circle lies below the black dotted line, it is impossible to satisfy the surge margin constraint for the given situation. Note that the fan always satisfies the lower limit, but LPC and HPC do not satisfy the lower limits for certain situations.

Note also that using LPSG more increases the fan and LPC surge margins, and using the HPSG more increases the HPC surge margin. Furthermore, for some split ranges for fan and LPC, surge margins increase as the total electrical power output increases.

We now consider the power split ranges that satisfy both fuel efficiency and surge margin constraints for the given thrust and total electrical power level. See Fig. 5.7.

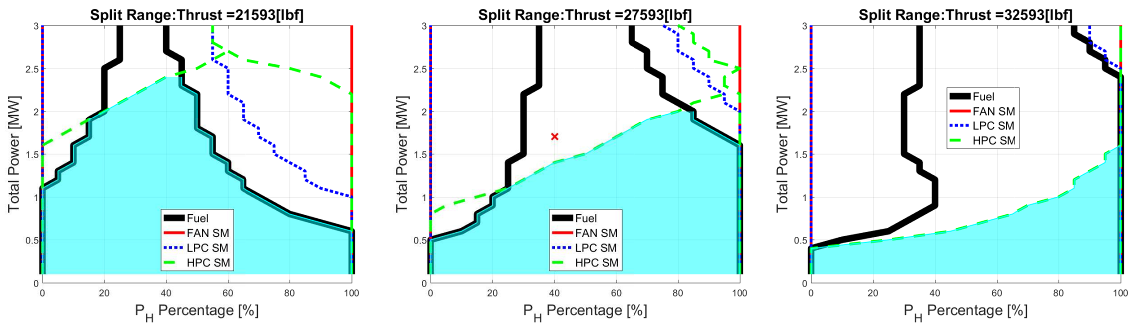

Figure 5.7: Fuel and surge margin optimal power split ranges.

Not all values of P_H percentage in the fuel optimal power split range satisfy the surge margin limits. For instance, for 27,593 [lbf] thrust and the total electrical power of 1.7 MW, P_H percentage of 40%, as indicated by the red cross, is within the fuel optimal range but it violates the HPC surge margin limit. The optimal power split range that takes into account the fuel efficiency constraints and surge margin limits is indicated in cyan in Fig. 5.7. The total electrical power output becomes more limited as the thrust increases, as expected. The optimal power split ranges for thrust varying between 21,593 and 32,593 [lbf] and total electrical power varying between 0 and 3 [MW] have been generated.

Note that for a given thrust and total electrical power, the optimal power split range equivalently prescribes lower and upper bounds for the power difference, $P_D =$

$P_H - P_L$, between HPSG and LPSG. Rather than using these values as constraints, in our MPC controller design, we choose to use both of these bounds ($P_{Dreq_{min}}$ and $P_{Dreq_{max}}$), respectively, as setpoints in the cost function for P_D. As a result, P_D is maintained in the range between these two setpoints as we have verified by simulations, and this design approach leads to good performance.

5.4.3 Energy Storage Elements Control Strategy

The energy storage SoC is constrained between 40% and 60%. These SoC constraints are treated as soft in the control design. The setpoint for the energy storage SoC is changed according to the following rule-based strategy:

- When thrust and load are decreased: track high SoC setpoint, which is 90% in our simulation case study – charge.
- When thrust and load are increased: track low SoC setpoint, which is 10% in our simulation case study – supply.
- When one is decreased and the other is maintained: track high SoC setpoint – charge.
- When one is increased and the other is maintained: track low SoC setpoint – supply.
- All other cases: track the setpoint corresponding to the mid-range between lower and upper limit – maintain desired SoC.

The basic idea behind these rules is to charge the energy storage if extra power is available, and discharge the energy storage if extra power is needed. Given a SoC setpoint of the energy storage j (SoC_{j_d}), the stored energy setpoint of the energy storage j can be computed based on Eq. (5.4) as follows:

$$E_{j_d} = SoC_{j_d} \times E_{j_{Max}}. \tag{5.13}$$

Thus, the stored energy setpoint in the MPC controller can be used instead of the SoC setpoint because the stored energy is one of the outputs of the linear model for our MPC controller design.

5.4.4 Rate-based MPC Controller Design

5.4.4.1 Scaled Model

To alleviate the effects of different order of magnitudes of the inputs and outputs for the MPC controller, the inputs and outputs of the linear model are scaled before controller design. We want to scale the inputs and outputs such that the maximum value of each element in the scaled inputs and outputs is one.

Let δu_s designate the vector of scaled inputs and $\delta u_{s_{max}}$ be the maximum value of the scaled inputs, so that each element in $\delta u_{s_{max}}$ is one. Let the vector of the maximum values of the inputs δu be given by $\delta u_{max} = [\delta u_{1_{max}}\ \delta u_{2_{max}}\ ...\ \delta u_{i_{max}}]'$. Then, the relationship between the inputs and scaled inputs is defined as

$$\delta u = S_u \delta u_s, \tag{5.14}$$

where S_u is the input scaling matrix,

$$S_u = \begin{bmatrix} \delta u_{1_{max}} & 0 & 0 & 0 \\ 0 & \delta u_{2_{max}} & 0 & 0 \\ 0 & 0 & ... & 0 \\ 0 & 0 & 0 & \delta u_{i_{max}} \end{bmatrix}. \tag{5.15}$$

Let δy_s be the vector of scaled outputs and $\delta y_{s_{max}}$ be the maximum value of the scaled outputs, so that each element in $\delta y_{s_{max}}$ is one. Assume that the maximum value of the outputs is known. Let δy be the outputs and $\delta y_{max} = [\delta y_{1_{max}}\ \delta y_{2_{max}}\ ...\ \delta y_{j_{max}}]'$ be the maximum value of the outputs. Then, the relationship between the outputs and scaled outputs is defined as

$$\delta y = S_y \delta y_s, \tag{5.16}$$

where S_y is the output scaling matrix,

$$S_y = \begin{bmatrix} \delta y_{1_{max}} & 0 & 0 & 0 \\ 0 & \delta y_{2_{max}} & 0 & 0 \\ 0 & 0 & ... & 0 \\ 0 & 0 & 0 & \delta y_{j_{max}} \end{bmatrix}. \tag{5.17}$$

Assume that the unscaled linear system from Eq. (5.12) has the following form:

$$\begin{aligned} \delta \dot{x} =& A'' \delta x + B'' \delta u \\ \delta y =& C'' \delta x + D'' \delta u. \end{aligned} \tag{5.18}$$

Substituting Eq. (5.14) and Eq. (5.16) into Eq. (5.18) yields

$$\begin{aligned} \delta \dot{x} =& A'' \delta x + B'' S_u \delta u_s \\ S_y \delta y_s =& C'' \delta x + D'' S_u \delta u_s. \end{aligned} \tag{5.19}$$

Then, the scaled system is

$$\begin{aligned} \delta \dot{x} =& A'' \delta x + \hat{B} \delta u_s \\ \delta y_s =& \hat{C} \delta x + \hat{D} \delta u_s, \end{aligned} \tag{5.20}$$

where $\hat{B} = B'' S_u$, $\hat{C} = {S_y}^{-1} C''$, and $\hat{D} = {S_y}^{-1} D'' S_u$. The model with the scaled inputs and outputs is used for control design.

5.4.4.2 Offset State

Before the rate-based model for MPC design is introduced, we describe how the nominal linear discrete-time model can be augmented with extra offset states to compensate for errors between linear model predictions and the response of the actual nonlinear system. The approach of compensating for model mismatch using offset states has also been used in other predictive control applications, such as for reference governors [100, 43]. For the design of rate-based MPC, a linear discrete-time model is needed. Let the discrete-time linear model of the system have the following form,

$$\begin{aligned} \delta x_{k+1} &= A'_d \delta x_k + B'_d \delta u_k, \\ \delta y_k &= C'_d \delta x_k + D'_d \delta u_k, \end{aligned} \tag{5.21}$$

where k indicates discrete time instant, and y_k denotes the output on which constraints are imposed. Suppose that the actual nonlinear system is given by

$$\begin{aligned} X_{k+1} &= f(X_k, U_k), \\ Y_k &= g(X_k). \end{aligned} \tag{5.22}$$

The offset state at the time instant t is defined as:

$$d_t = Y_t - (\delta y_t + y_{no}), \tag{5.23}$$

where δy_t is the vector of outputs of the linearized model (deviations from the nominal values) at the time instant t and y_{no} is the vector of nominal values of the output at which the model is linearized. We assume that the measurements or accurate estimates of Y_t are available so that the current value of the offset state d_t can be computed. Then, the linear prediction model is given by

$$\begin{aligned} \delta x_{k+1|t} &= A'_d \delta x_{k|t} + B'_d \delta u_{k|t}, \\ d_{k+1|t} &= d_{k|t}, \\ \delta y_{k|t} &= C'_d \delta x_{k|t} + D'_d \delta u_{k|t} + d_{k|t}, \end{aligned} \tag{5.24}$$

where standard notation in predictive control is used to designate predictions, e.g., $\delta x_{k|t}$ is the predicted state k steps ahead when the prediction is made at the time instant t. In the sequel, this approach is used for handling surge constraints and hence we assume, motivated by existing literature, see, e.g., [26], that accurate estimates or measurements of surge margins are available in the gas turbine engine control strategy to be able to compute $d_{0|t} = d_t$.

5.4.4.3 Rate-based MPC

The design process of the rate-based MPC controller is now described. The states of the linear model used for prediction are assumed to be available from measurements and appropriately designed estimators. The rate-based MPC design described in this section is for the system configuration with a single energy storage element and two surge margin offset states. Other system configurations are handled similarly.

The discrete-time model is obtained using a sampling period of 0.04 sec based on the scaled input-output model in Eq. (5.20). A rate-based MPC controller can be designed to perform setpoint tracking based on the discrete-time prediction model shown, without extra offset states, as

$$\delta x_{k+1} = A_d \delta x_k + B_d \delta u_k, \;\; \delta y_k = C_d \delta x_k + D_d \delta u_k, \tag{5.25}$$

where $A_d^{6\times6}$, $B_d^{6\times4}$, $C_d^{5\times6}$, $D_d^{5\times4}$, and $\delta y_k = [\delta F_g \;\; \delta P_T \;\; \delta P_D \;\; \delta P_D \;\; E_j]^T$. The control objective is to follow a requested command (setpoint) r where $r = [\delta Fg_{req} \;\; \delta P_{Treq}$

$\delta P_{Dreq_{max}} \quad \delta P_{Dreq_{min}} \quad E_{jd}]^T$, that is, follow thrust requests, total electrical power requests, optimal maximum power difference requests, optimal minimum power difference requests, and stored energy requests, respectively. Then, the state and control increments are defined as

$$\Delta x_k = \delta x_{k+1} - \delta x_k,\ \Delta u_k = \delta u_{k+1} - \delta u_k, \tag{5.26}$$

and the error between outputs (y_k) and setpoints (r) is defined as

$$e_k = C_d \delta x_k + D_d \delta u_k - r. \tag{5.27}$$

Then,

$$\begin{aligned} \Delta x_{k+1} &= A_d \Delta x_k + B_d \Delta u_k, \\ e_{k+1} &= C_d \Delta x_k + D_d \Delta u_k + e_k, \\ \delta x_{k+1} &= \delta x_k + \Delta x_k, \\ \delta u_{k+1} &= \delta u_k + \Delta u_k. \end{aligned} \tag{5.28}$$

Eq. (5.28) can be extended with two surge margin offset states and two compensated surge margin states as described in Sec. 5.4.4.2. The extended linear prediction model is as follows:

$$\begin{aligned} \Delta x_{k+1} &= A_d \Delta x_k + B_d \Delta u_k, \\ e_{k+1} &= C_d \Delta x_k + D_d \Delta u_k + e_k, \\ \delta x_{k+1} &= \delta x_k + \Delta x_k, \\ \delta u_{k+1} &= \delta u_k + \Delta u_k, \\ d_{k+1} &= d_k, \\ \delta \bar{x}_{k+1} &= F \delta x_{k+1} + d_{k+1} = F \delta x_k + F \Delta x_k + d_k, \end{aligned} \tag{5.29}$$

where d_k is the 2×1 surge margin offset states vector, $\delta \bar{x}_{k+1}$ is the 2×1 compensated surge margin deviations vector, and $F = [0_{2\times 4}\ I_{2\times 2}]$. The cost function to be minimized is given by

$$J_N = \sum_{k=0}^{N-1} e_{k|t}^T Q e_{k|t} + \Delta u_{k|t}^T R \Delta u_{k|t},$$

$$\begin{aligned} \text{subject to the constraints: } & \delta x_{min} \leq \delta x_{k|t} \leq \delta x_{max},\ k = 0, \cdots, N, \\ & \delta u_{min} \leq \delta u_{k|t} \leq \delta u_{max},\ k = 0, \cdots, N-1, \\ & \Delta u_{min} \leq \Delta u_{k|t} \leq \Delta u_{max},\ k = 0, \cdots, N-1, \end{aligned} \tag{5.30}$$

where N is the prediction horizon, Q is a 5×5 diagonal weight matrix associated with the five errors, R is a 4×4 diagonal weight matrix associated with the four inputs, $e_{k|t}$ is the predicted error k steps ahead when the prediction is made at time instant t, $\delta u_{k|t}$ is is the predicted input k steps ahead when the prediction is made at time instant t, δx_{min} and δx_{max} designate state bounds, and $\delta u_{min}, \delta u_{max}, \Delta u_{min}$ and Δu_{max} designate the bounds on the control inputs and their time rates of change.

Note that the cost function is constructed to penalize the deviation of power difference between the two generators (P_D) from the maximum power difference setpoint ($P_{Dreq_{max}}$) and the minimum power difference setpoint ($P_{Dreq_{min}}$), where these setpoints are computed from optimal power split ranges. The same weights are used for both tracking errors. This strategy maintains P_D in between the two setpoints and hence within/in the middle of the optimal power split range.

The above tracking MPC formulation can be re-written as a standard MPC problem (to which standard MPC solvers are applicable) for an extended system with a larger state vector,

$$x_{k|t}^{ext} = \begin{bmatrix} \Delta x_{k|t}^T & e_{k|t}^T & \delta x_{k|t}^T & \delta u_{k|t}^T & d_{k|t}^T & \delta \bar{x}_{k|t}^T \end{bmatrix}^T, \tag{5.31}$$

and the extended state prediction model given by

$$x_{k+1|t}^{ext} = \begin{bmatrix} A_d & 0 & 0 & 0 & 0 & 0 \\ C_d & I_{5\times 5} & 0 & 0 & 0 & 0 \\ I_{6\times 6} & 0 & I_{6\times 6} & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{4\times 4} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{2\times 2} & 0 \\ F & 0 & F & 0 & I_{2\times 2} & 0 \end{bmatrix} x_{k|t}^{ext} + \begin{bmatrix} B_d \\ D_d \\ 0 \\ I_{4\times 4} \\ 0 \\ 0 \end{bmatrix} \Delta u_{k|t}. \tag{5.32}$$

For this extended system, the state penalty matrix has the form

$$Q_{ext} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & Q & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{5.33}$$

and the control penalty matrix is $R_{ext} = R$. Two choices of prediction horizon and sampling period are considered: N=100 with a sampling period of 0.04 sec (which corresponds to 4 sec of prediction) and N=30 with a sampling period of 0.12 sec (which corresponds to 3.6 sec of prediction). The MPT toolbox [55] is used to implement and simulate our MPC controller. Hard constraints are imposed on power output of the generators to be positive and power limit to/from the energy storage elements, and soft constraints are imposed on surge margins and stored energy of the energy storage elements.

5.4.5 Multiple MPC Controllers

The rate-based MPC controller is based on a single linear system obtained as a linearization of the nonlinear model at 27,593 [lbf] of thrust and 0 electrical load. If the system operates far from this nominal operating point, model inaccuracies may lead to poor closed loop performance. The standard approach to address this issue [77, 28, 46], sometimes called switched MPC, is to design a set of linear MPC controllers based on linear models at several operating points, and then switch between the corresponding MPC controllers depending, in our case, on the engine thrust level. The switching process can be summarized as follows:

1. If the current operating point is different from the previous operating point, go to step 3. Otherwise go to step 2.
2. Generate control input using the current controller, then return to step 1.
3. Switch the controller and initialize the previous linear state as follows:

$$\delta x_{old} = 0. \tag{5.34}$$

4. Update the previous input as follows:

$$\delta u_{old} = \delta u_{old} - (u_{n0} - u_{old0}), \tag{5.35}$$

   where u_{n0} is the nominal input at the new operating point and u_{old0} is the nominal input at the previous operating point.

5. Update the current linear state as follows:

$$\delta x = A_d \delta x_{old} + B_d \delta u_{old}, \tag{5.36}$$

where A_d and B_d are the discrete linear system matrices at the new operating point.

6. Compute the input for the MPC controller as follows:

$$x_{cont} = [(\delta x - \delta x_{old})^T \ e^T \ \delta x^T \ \delta u_{old}^T \ d^T \ \delta x^T + d^T]^T, \tag{5.37}$$

   where e is the measured error and d is offset states. Then, generate the control input using the current controller, and return to step 1.

Our switching MPC design was based on 10 operating points corresponding to thrust levels 20,593, 21,593, 22,593, 23,593, 24,593, 25,593, 26,593, 27,593, 28,593 and 29,593 [lbf]. At each operating point, the linearized model was generated using identification techniques as in Sec. 5.2.4.

Table 5.1: Agreement between the validation data and identified linear model.

	Operating Point Determined by Thrust Level (lbf)									
	20,593	21,593	22,593	23,593	24,593	25,593	26,593	27,593	28,593	29,593
HPS Speed	94.11%	91.89%	92.48%	88.54%	82.82%	82.19%	83.38%	81.34%	80.60%	89.54%
LPS Speed	92.66%	92.73%	82.14%	80.39%	79.00%	83.50%	81.36%	80.24%	82.32%	87.54%
Thrust	93.72%	93.21%	89.14%	84.78%	87.11%	82.69%	83.00%	81.41%	82.10	88.87%
LPC SM	90.44%	86.29%	84.46%	76.60%	72.83%	64.32%	64.97%	63.80%	69.89%	79.59%
HPC SM	90.02%	92.43%	52.71%	42.85%	51.72%	68.44%	40.84%	82.62%	43.69%	89.94%
Average	92.99%	91.31%	80.19%	74.63%	74.70%	76.23%	71.11%	77.88%	71.72%	87.09%

The agreements between the validation data and identified linear models have been computed for each operating point as described in Sec. 5.2.4.1. See Table 5.1. Because the LPC and HPC surge margins behaviors are highly nonlinear as shown in Fig. 5.3, the LPC and HPC surge margins agreements are relatively poor compared to the other agreements. However, these inaccuracies can be handled by using the extra offset state as described in Sec. 5.4.4.2. The 10 MPC controllers used in switched MPC design were generated based on these linearized models and the same weights.

5.5 Simulations and Results

The results of different simulation case studies are reported in this section. Firstly, uncoordinated Linear Quadratic Regulators (LQR), integrated LQR and integrated MPC are compared to show the benefits of the integrated control and of the MPC. Secondly, the responses with different energy storage elements are compared, and the benefits of adding energy storage elements to the system are illustrated. Thirdly, LQR, MPC, MPC with surge margin offset states (offset MPC) and 10 MPC (Multiple Model Predictive Control (MMPC)) control designs are compared. The MPC controllers with offset state for the systems configurations with and without energy storage elements are also compared.

The control objective is to satisfy the surge margin constraints, maintain the requested thrust level, and supply the requested electrical power during a 90-sec simulation. All the simulations start from steady-state with $Fg = 27,593\ lbf$, LPC surge margin = 44.7%, HPC surge margin = 17.3%, $FAR = 0.0187$, and $P_{Hreq} = P_{Lreq} = P_{jreq} = P_T = 0\ MW$. The initial energy storage SoC is 50%, and the desired SoC range is between 40 and 60%. The state and input constraints are summarized in Table 5.2. Since LQR controllers do not enforce constraints, no constraints are defined for them.

Table 5.2: Constraints for LQR and MPC controllers.

			LQR	MPC
Input Constraints	FAR		N/A	±inf
	P_H	[KW]	N/A	$0 \leq$
	P_L	[KW]	N/A	$0 \leq$
	$\dot{FAR}$	[/s]	N/A	±0.0005
	$\dot{P}_H$	[KW/s]	N/A	±1,000
	$\dot{P}_L$	[KW/s]	N/A	±1,000
State Constraints	SM_{FAN}	[%]	N/A	$15 \leq$
	SM_{LPC}	[%]	N/A	$20 \leq$
	SM_{HPC}	[%]	N/A	$14 \leq$

The charge/discharge rate constraints of the energy storage elements vary based on their types, so these constraints are as indicated for each simulation.

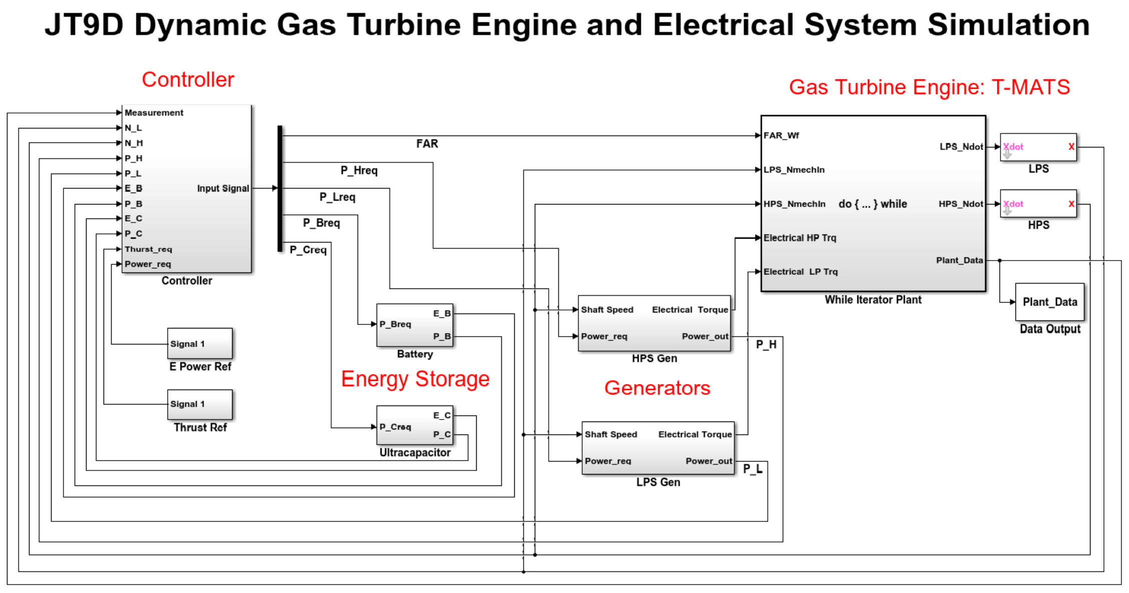

Figure 5.8: Simulink model for the closed-loop system with the offset MPC.

All simulations are performed on the fully nonlinear model of the system. The Simulink model of the closed-loop system with the offset MPC controller and the energy storage elements is shown in Fig. 5.8.

5.5.1 Performance Metrics

Performance metrics have been defined to compare different controllers. The first metric is the average thrust deviation from the setpoint, Fg_{AvgDev}, which reflects the thrust tracking performance, and is defined by

$$Fg_{AvgDev} = \frac{\sum |Fg_{ref} - Fg|}{n_t} \text{ or } \frac{\int_0^{t_d} |Fg_{ref} - Fg|}{t_d}, \tag{5.38}$$

where Fg_{ref} is the thrust setpoint, Fg is the thrust, n_t is the number of samples, and t_d is the total simulation duration. A smaller value of Fg_{AvgDev} indicates better thrust request tracking.

The second metric is the average total electrical power deviation from the setpoint, $P_{T_{AvgDev}}$, which reflects the performance in supplying the requested total electrical power. This metric is defined as

$$P_{T_{AvgDev}} = \frac{\sum \left|P_{T_{ref}} - P_T\right|}{n_t} \text{ or } \frac{\int_0^{t_d} \left|P_{T_{ref}} - P_T\right|}{t_d}, \tag{5.39}$$

where $P_{T_{ref}}$ is the total electrical power setpoint (i.e., the sum of required electrical loads), and P_T is the total electrical power generated by the system. A smaller value of $P_{T_{AvgDev}}$ indicates better total electrical power tracking, i.e., better supply of the electrical loads.

The next set of metrics is introduced to characterize the surge margin violations. The metrics are: the number of surge margin violations (n_{smv}), the duration of the i^{th} violation ($t^i_{d_{smv}}$), and the maximum amount of the i^{th} violation (SM^i_{MaxV}).

The final metric is the total fuel consumption, w_f. A smaller value of w_f indicates better fuel efficiency.

5.5.2 Uncoordinated and Coordinated Control

In this section, three different controllers are compared: uncoordinated LQR, integrated LQR, and integrated MPC. For the purpose of this comparison, the system without the energy storage elements is considered. The uncoordinated LQR controller only adjusts FAR for the engine, while the generator power requests are managed according to a simple strategy of the form,

$$\begin{aligned} P_{Hreq} &= \tfrac{P_{Treq}+P_{Dreq}}{2}, \\ P_{Lreq} &= \tfrac{P_{Treq}-P_{Dreq}}{2}. \end{aligned} \tag{5.40}$$

Table 5.3: Parameters for LQR and MPC controllers.

		Uncoordinated LQR	Integrated LQR	Integrated MPC
Sampling Time	[s]	0.04	0.04	0.04
Prediction Horizon	[steps]	Inf	Inf	30
Constraint Horizon	[steps]	N/A	N/A	30
Control Horizon	[steps]	Inf	Inf	10

In the integrated LQR and MPC controllers, a single controller is used for the whole system to provide coordinated control of three inputs. Each choice of the controller was tuned for best performance. The controller parameters are shown in Table 5.3.

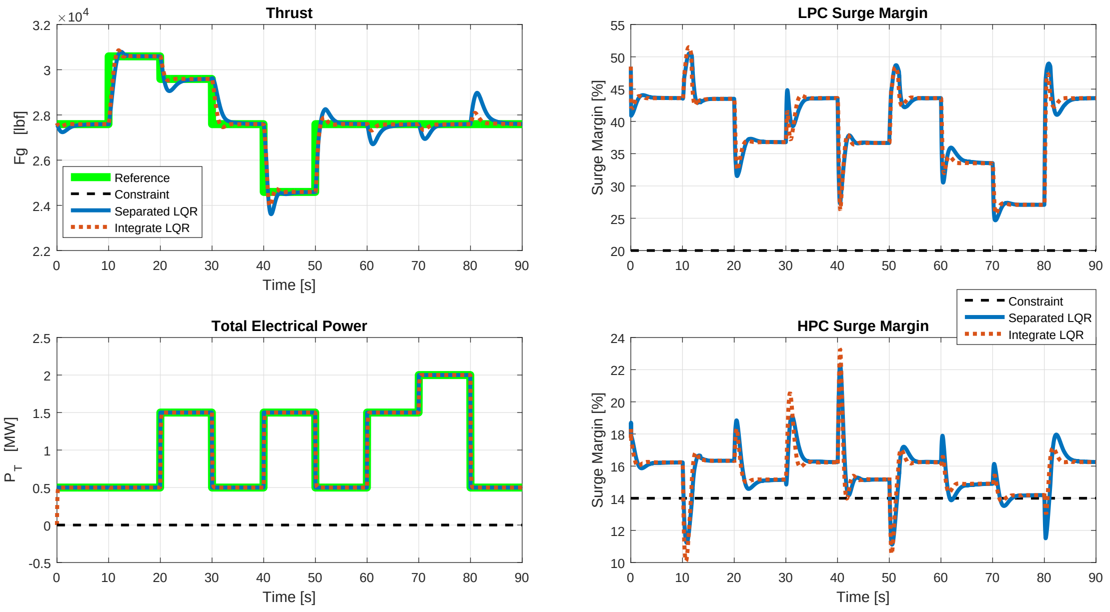

Figure 5.9: Comparison of uncoordinated LQR and integrated LQR controllers.

The simulation results of uncoordinated LQR and integrated LQR controllers are shown in Fig. 5.9. As shown in the left subfigures, both controllers accurately track the total electrical power setpoint. However, the uncoordinated LQR controller yields larger thrust deviations when the step change of electrical power occurs as it does not account for the interactions between the generators and the gas turbine engine. As expected, both controllers violate surge margin constraints.

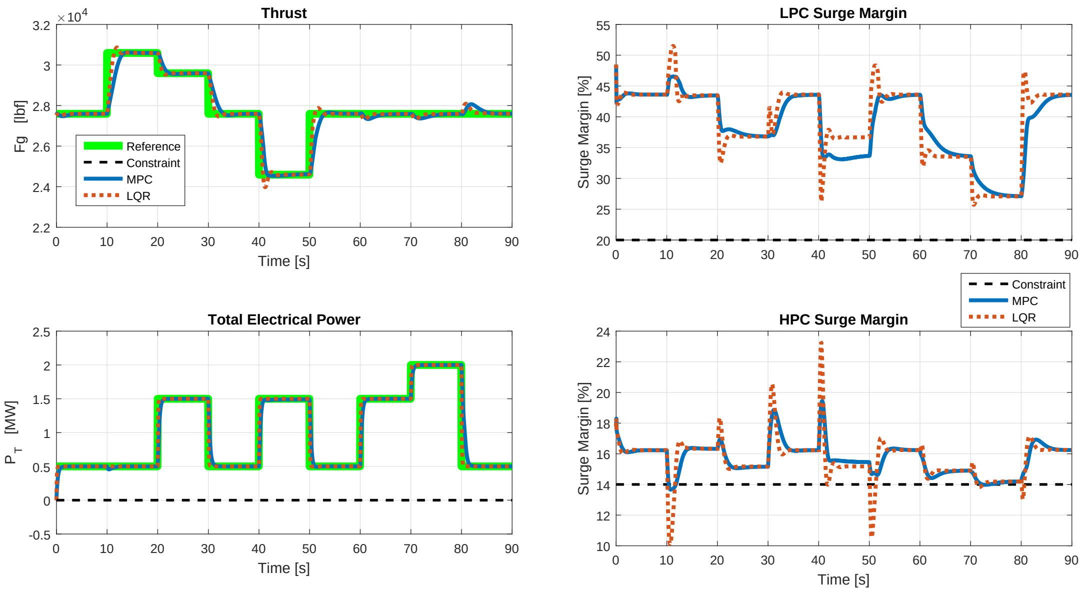

Figure 5.10: Comparison of integrated LQR and MPC controllers.

The simulation results of the integrated LQR and integrated MPC controllers are shown in Fig. 5.10. As shown in the right subfigures, the integrated MPC controller satisfies the soft surge margin constraints, except for a few small violations, while the integrated LQR controller does not. Note that the tracking of thrust and total electrical power is worse for the integrated MPC than for the integrated LQR controller; however, unlike LQR, the former enforces the constraints, see Fig. 5.11.

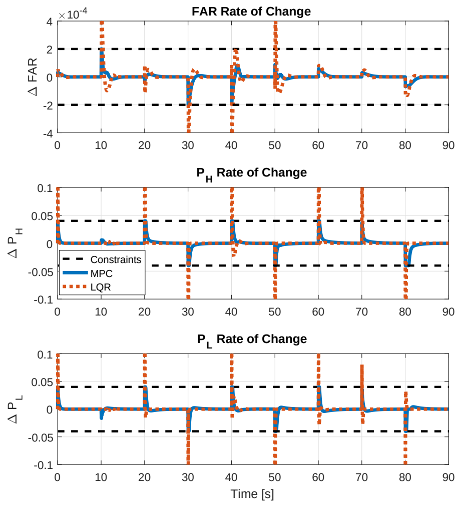

Figure 5.11: Input constraint comparison of integrated LQR and MPC controllers.

Table 5.4: Comparison of uncoordinated LQR, integrated LQR, and integrated MPC.

		Uncoordinated LQR	Integrated LQR	Integrated MPC
W_f	[Kg]	84.74	84.77	84.59
Fg_{AvgDev}	[lbf]	236.13	109.05	191.79
$P_{T_{AvgDev}}$	[KW]	0	7.42	32.48
n_{smv}	[times]	5	5	2
$t_{d_{smv}}$	[s]	1.6 / 1.36 / 0.68 / 2.28 / 0.92	1.44 / 0.12 / 1.16 / 1.16 / 0.48	1.36 / 1.32
SM_{MaxV}	[%]	2.78 / 2.89 / 0.13 / 0.48 / 2.49	3.90 / 0.02 / 3.47 / 0.17 / 0.97	0.34 / 0.03

The performance metrics for the three controllers are compared in Table 5.4. The uncoordinated LQR controller yields the best electrical power tracking performance, but the worst thrust tracking performance. The integrated LQR controller yields the best thrust tracking performance and good electrical power tracking performance. However, both LQR controllers violate the HPC surge margin constraint five times, sometimes by a large amount (2.89% for the uncoordinated LQR and 3.9% for the integrated LQR at time instants 50.52 sec and 10.68 sec, respectively), while the integrated MPC controller only violates these constraints twice by very small amounts.

To illustrate the advantages of MPC over LQR, note that tuning the LQR controller less aggressively could remove the surge margin violations, but, at the same time, the thrust and total electrical power tracking will be slower for all the transients, even for small transients, for which there is no danger of surge margin violations. Examples of different tunings of the integrated LQR controller and comparison with integrated MPC are shown in Fig.5.12.

As indicated in the figures, tuning the integrated LQR controller less aggressively, indicated by LQR1 in Fig. 5.12, reduces the surge margin violations. However, it stills yields more surge margin violations, and furthermore, worse thrust tracking performance than the integrated MPC. If the integrated LQR controller is tuned further (less aggressively), indicated by LQR2 in the figures, most of the surge margin violations disappear, but thrust and total electrical power tracking are poor. Meanwhile, the MPC controller can provide aggressive thrust and total electrical power track-

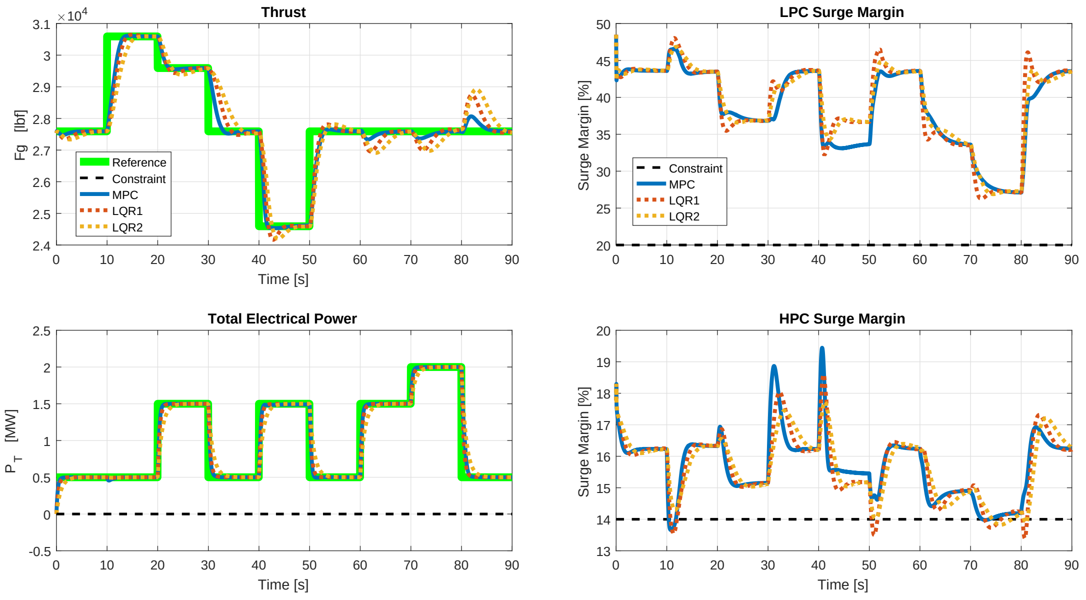

Figure 5.12: Comparison of integrated LQRs and MPC controllers.

ing when there is no danger of surge margin constraint violation, and less aggressive tracking when the surge margin constraints are active. Furthermore, in addition to the surge margins, there are other constraints handled by MPC controller (e.g., positive power limit, charge and discharge rate of the energy storage elements, etc.) that the LQR controller is not designed to handle. In the subsequent sections, the uncoordinated LQR controller design is no longer considered, and only the integrated controllers are focused on.

5.5.3 Different Energy Storage Elements

Table 5.5: Specification of battery cell and ultracapacitor cell.

	AMP20 Battery cell	K2 Ultracapacitor cell
Weight	496 g	520 g
Stored Energy, nominal	65 Wh	4 Wh
Discharge Power, nominal	1.2 KW	4.4 KW (max 9.4 KW)
Voltage, nominal	3.3 V	2.85 V

In this section, the system responses with different energy storage elements are compared, and the benefits of adding energy storage elements are illustrated. Batteries and ultracapacitors are chosen as the energy storage elements. The specifications of the chosen battery cells [2] and ultracapacitor cells [1] are listed in Table 5.5.

Table 5.6: Specifications of three different energy storage element configurations.

			Battery-Ultracapacitor Pack (Bat-Ucap Pack)	
	Battery-Pack (Bat Pack)	Ultracapacitor-Pack (Ucap Pack)	Battery-Pack (Bat Pack)	Ultracapacitor-Pack (Ucap Pack)
Number of cells	77 cells	88 cells	38 cells	44 cells
Weight	39 kg	46 Kg	19 Kg	23 Kg
Volume	20.3 L	36 L	10 L	18 L
Stored Energy, nominal	5 KWh	0.35 KWh	2.4 KWh	0.176 KWh
Voltage, nominal	$\approx$ 250 V	$\approx$ 250 V	$\approx$ 125 V	$\approx$ 125 V
Discharge Rate, nominal	92 KW	389 KW (max 824 KW)	45 KW	193 KW (max 413 KW)

Based on the specifications, three different energy storage element configurations are considered: a battery pack, an ultracapacitor pack, and a battery-ultracapacitor pack. All the energy storage elements are limited to less than 50 Kg and 40 L considering limited space on aircraft. Note that currently, relatively small stored energy is considered compared to our electrical load requests. The specifications of the energy storage elements are shown in Table 5.6.

Table 5.7: Parameters and energy storage element input constraints for MPC.

			MPC				
			Without	Bat Pack	UCap Pack	Bat-Ucap Pack	
						Bat Pack	UCap Pack
Controller Parameters	Sampling Time	[s]	0.04	0.04	0.04	0.04	0.04
	Prediction Horizon	[steps]	100	100	100	100	100
	Constraints Horizon	[steps]	100	100	100	100	100
	Control Horizon	[steps]	30	30	30	30	30
Input Constraints	P_j	[KW]	N/A	$\pm$100	$\pm$800	$\pm$50	$\pm$400
	$\dot{P}_j$	[KW/s]	N/A	$\pm$inf	$\pm$inf	$\pm$inf	$\pm$inf

There are clearly differences between the battery pack and the ultracapacitor pack in terms of the stored energy and the discharge rate. The battery pack has much larger stored energy than the ultracapacitor pack, but the ultracapacitor pack has much faster discharge rate than the battery pack. The battery-ultracapacitor packs can take advantage of both characteristics. Based on the specifications in Table 5.6, the controller parameters and the energy storage element input constraints (charge/discharge rate constraints) are defined as shown in Table 5.7.

All of the MPC controllers designed for systems with three different storage element configurations use the same controller parameters and constraints except for the constraints on P_j that are determined based on the discharge rate of the energy storage elements. Constraints on $\dot{P}_j$ are not considered. The simulation results of MPC with the battery pack and the ultracapacitor pack are shown in Fig. 5.13.

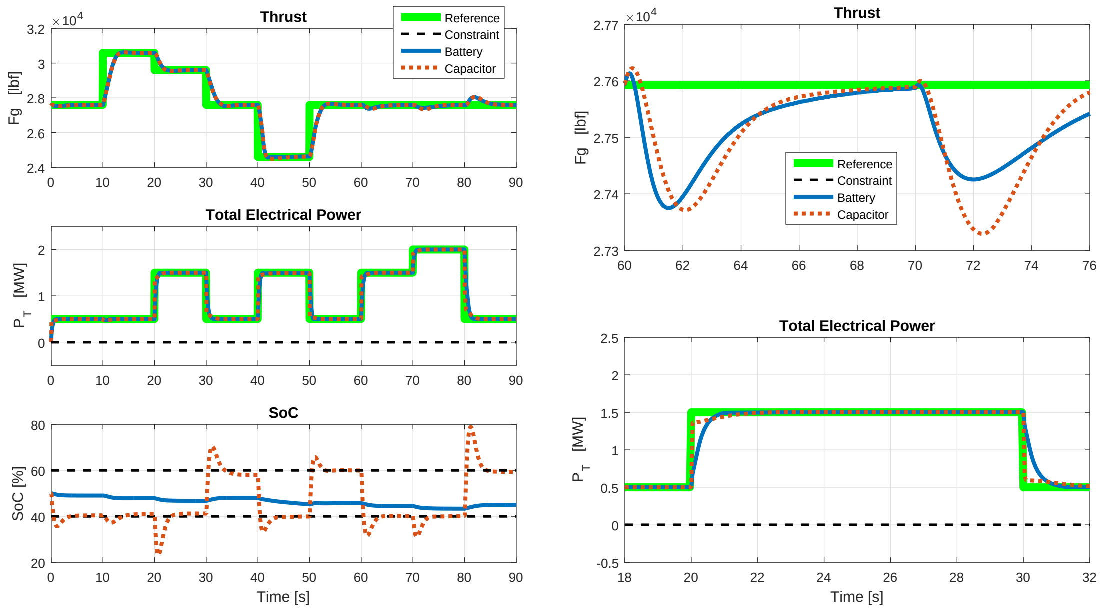

Figure 5.13: Comparison of battery and ultracapacitor packs.

There are clear differences in the responses observed for the two types of energy storage elements. The SoC of the ultracapacitor pack varies more than the SoC of the battery pack because the ultracapacitor pack has much smaller stored energy than the battery pack, while the ultracapacitor pack is much faster than the battery pack, so it can supply the electrical loads very quickly, see the right-bottom subfigure, which shows the time history of the total electrical power in the time interval between 18 and 32 sec. However, due to limited stored energy, the ultracapacitor cannot supply the electrical loads for a long time; instead it needs to be recharged to recover a SoC in the 40 to 60% range as shown in the left-bottom SoC subfigure in the time interval between 20 and 25 sec. Meanwhile, the battery pack allows for better thrust command tracking as shown in the right-upper subfigure in Fig. 5.13, that represents the trajectory of thrust in the time interval between 60 and 75 sec. This is reasonable given the large stored energy in the battery pack. The battery pack can deliver electrical power for a long time, which helps the gas turbine engine to use power for thrust generation instead of supplying the generators to satisfy the loads. In addition, the battery pack reduces surge margin violations, as shown in Table 5.8. Thus, for faster electrical loads supplying, the ultracapacitor pack appears to be a suitable energy storage choice; however, for faster thrust responses and stable gas turbine engine operation, the battery pack is preferred.

We next compare cases without and with the battery-ultracapacitor pack that take advantage of the characteristics of both types of energy storage elements. The simulation results are shown in Fig. 5.14.

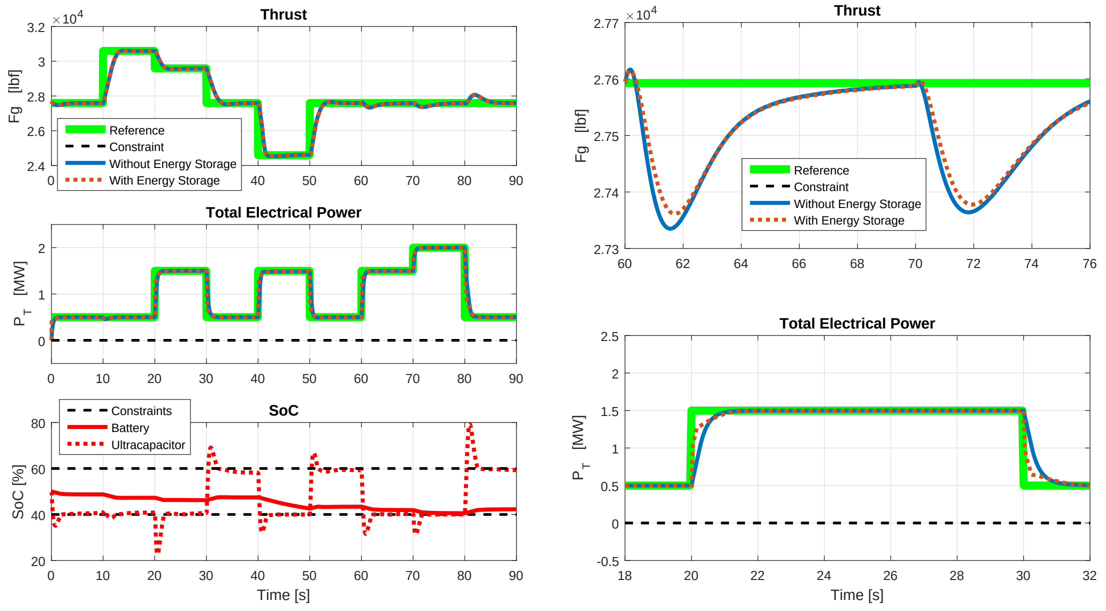

Figure 5.14: Comparison of controller without and with energy storage elements.

As shown in the right subfigures in Fig. 5.14, with the energy storage element, the controller has better thrust and electrical power tracking. The comparison of the performance of all the cases is shown in Table 5.8. All energy storage element cases yield better performance than the case without the energy storage element except for the second surge margin violation with the ultracapacitor pack. The second surge margin violation for the ultracapacitor pack is longer and larger than in the case without energy storage elements. This violation likely occurs because the ultracapacitor pack needs to be charged frequently, which requires the gas turbine engine to provide more output than in the case without the ultracapacitor. Thus, using the battery-ultracapacitor pack appears to be the preferred choice from the perspective of system response to thrust commands and electrical loads, and if the impact on weight, packaging and cost is not considered.

Table 5.8: Comparison of the systems with different energy storage elements.

		MPC			
		Without	Bat Pack	UCap Pack	Bat-Ucap Pack
W_f	[Kg]	84.59	84.56	84.6	84.57
Fg_{AvgDev}	[lbf]	191.79	187.17	191.87	188.84
$P_{T_{AvgDev}}$	[KW]	32.48	26.69	15.02	19.32
n_{smv}	[times]	2	1	2	2
$t_{d_{smv}}$	[s]	1.36 / 1.32	1.32	1.36 / 1.72	1.32 / 0.76
SM_{MaxV}	[%]	0.34 / 0.03	0.34	0.34 / 0.06	0.34 / 0.01

Note that the energy storage elements are beneficial, based on our simulation results, despite the fact that their stored energy is limited in this study. Even more substantial benefits are expected for the energy storage elements with larger stored energy in future MEAs.

5.5.4 LQR, MPC, Offset MPC, and MMPC Controllers

In this section, LQR, MPC, offset MPC and MMPC controllers are compared. The MMPC controller based on linearizations at multiple (10 in our case) operating points and the MPC controller with extra offset states are considered as potential approaches to better deal with the nonlinearities. For the purpose of quantifying potential benefits of these design steps, the energy storage elements are not included in the analysis, and a broader range of thrust profiles is used compared to the previous simulations. The controller parameters are shown in Table 5.9.

Table 5.9: Parameters for LQR, MPC, offset MPC and MMPC controllers.

		LQR	MPC	MMPC	Offset MPC
Sampling Time	[s]	0.12	0.12	0.12	0.12
Prediction Horizon	[steps]	Inf	30	30	30
Constraint Horizon	[steps]	N/A	30	30	30
Control Horizon	[steps]	Inf	10	10	10

For all the controllers, the same parameters and constraints are used. The closed-loop performance comparison is shown in Table 5.10.

Table 5.10: Comparison of LQR, MPC, MMPC, and offset MPC controllers.

		LQR	MPC	MMPC	Offset MPC
W_f	[Kg]	75.34	74.47	75.13	74.28
Fg_{AvgDev}	[lbf]	170.06	719.72	497.93	718.93
$P_{T_{AvgDev}}$	[KW]	7.50	101.14	92.18	101.14
n_{smv}	[times]	4	1	1	0
$t_{d_{smv}}$	[s]	1.44 / 1 / 5.4 / 0.48	5.36	7.76	0
SM_{MaxV}	[%]	3.9 / 3.86 / 1.24 / 0.97	1.06	1.62	0

As expected, the LQR controller yields the best thrust and electrical load tracking, but it violates the surge margin constraints four times with a maximum of 3.9%. It also consumes the largest fuel amount. The MPC controller is able to reduce the surge margin violations; nevertheless it violates the surge margin constraint once at 5.36 sec with a maximum violation of 1.06%. This violation is likely due to discrepancy between the prediction model and the actual nonlinear plant behavior. The MMPC design yields better thrust and electrical loads tracking than the MPC controller, but the surge margin constraint violation is longer and has larger magnitude than the

MPC controller, which is likely due to surge margin prediction being insufficiently accurate. The offset MPC design yields very similar tracking performance to that of the MPC controller, and has no violation of the surge margin constraint. The simulation results of MPC, MMPC, and offset MPC controllers are shown in Fig. 5.15.

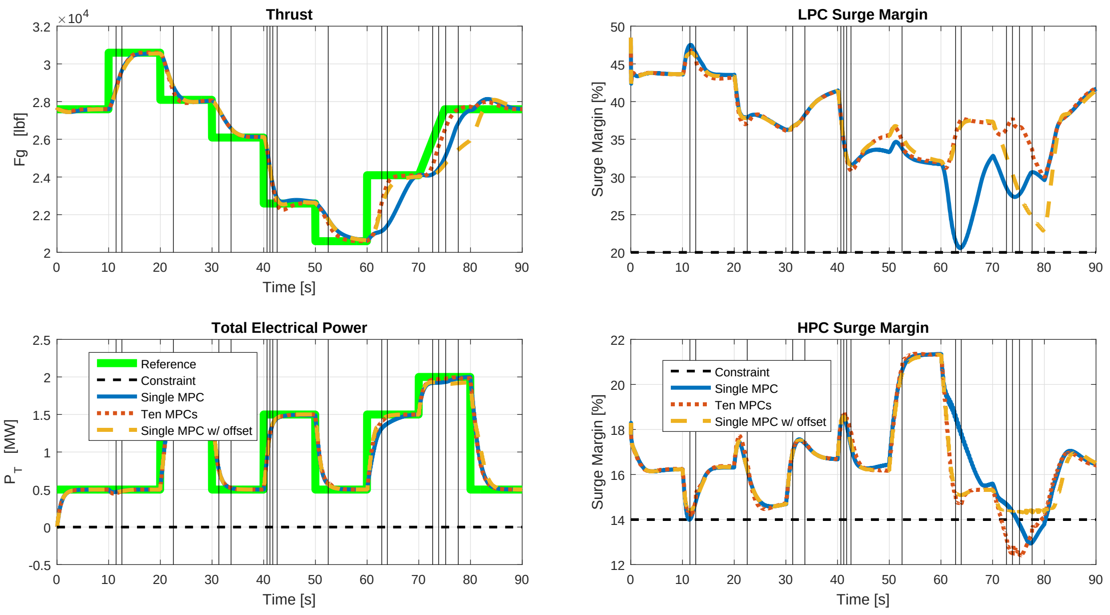

Figure 5.15: Comparison of MPC, MMPC, and offset MPC controllers.

The black vertical lines indicate the switching time instants for MMPC controllers. As observed, switching does not cause improper behaviors of the system. For most of the simulation, all three controllers yield similar results, but the major differences can be found in the time interval between 60 and 80 sec. Specifically, between 60 and 70 sec, the MPC controller assumes that it does not have available HPC surge margin because of inaccurate surge margin estimation due to being far from the operating point. Thus, it does not track the thrust setpoint aggressively, while the MMPC and offset MPC controllers are able to correctly account for the available HPC surge margin, and hence they track the thrust setpoint faster than the MPC controller. In the time interval between 70 and 80 sec, the MPC and MMPC controllers incorrectly assess that there is available HPC surge margin, so they track the thrust setpoint aggressively, but the offset MPC assesses that there is no available surge margin, so it tracks the thrust setpoint slowly to satisfy the HPC surge margin constraint, and uses the available LPC surge margin. Thus, the offset MPC performs best in this simulation.

To confirm that the offset MPC is a good design choice, an energy storage element, the battery-ultracapacitor pack, is added to the offset MPC controller, and the cases without and with the energy storage element are compared to each other. The controller parameters and the constraints can be found in Tables 5.2, 5.7 and 5.9. The performance comparison is shown in Table 5.11.

Table 5.11: Comparison of offset MPC with and without energy storage elements.

		Offest MPC	
		Without	With Bat-Ucap Pack
W_f	[Kg]	74.28	74.41
Fg_{AvgDev}	[lbf]	718.93	672.91
$P_{T_{AvgDev}}$	[KW]	101.14	71.35
n_{smv}	[times]	0	0
$t_{d_{smv}}$	[s]	0	0
SM_{MaxV}	[%]	0	0

As shown in Table 5.11, both thrust and total electrical power tracking performance are improved with the addition of the energy storage, especially in terms of the total electrical power tracking performance. The fuel consumption is increased, as a penalty for better tracking performance, but the increased amount is relatively small. The surge margin constraints are perfectly satisfied for both controllers. The simulation results of offset MPC controllers with and without energy storage elements are shown in Fig. 5.16.

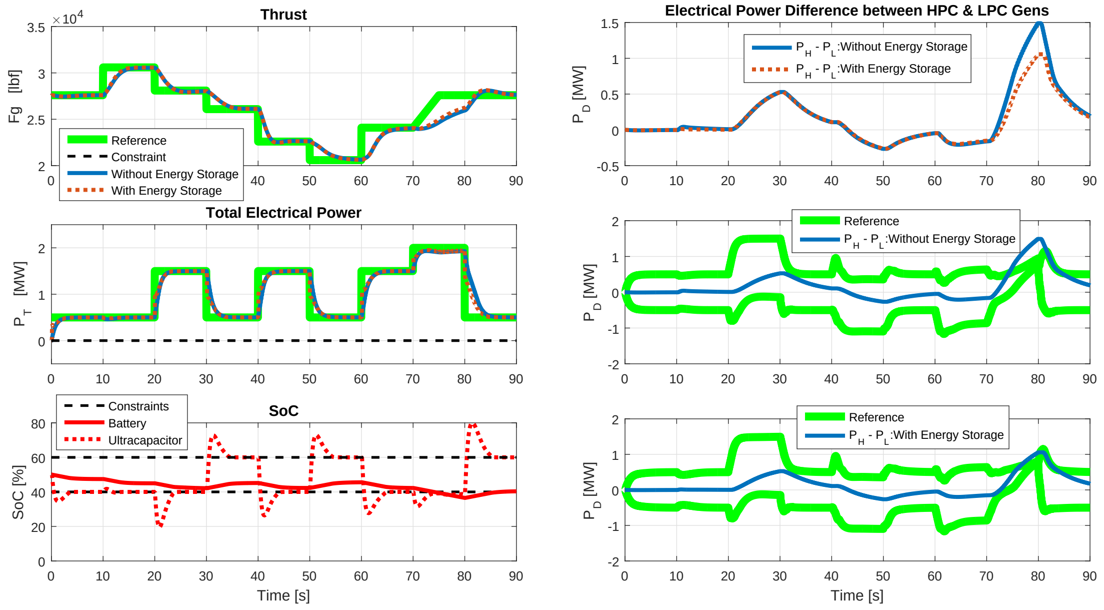

Figure 5.16: Comparison of offset MPC with and without energy storage elements.

As the left subfigures in Fig. 5.16 show, offset MPC with energy storage elements shows better thrust and total electrical power tracking. Both SoCs, especially for the ultracapacitor pack, violate the SoC constraint a small number of times to deal with transient thrust and electrical power changes, but they quickly recover to their constrained levels. As shown in the right subfigures, for both controllers, the power difference between the two generators stays within the optimal power split ranges for most of the time, which corresponds to safe and efficient operation.

5.5.5 Offset MPC with and without Sensor Noise

In this section, sensor noise is added to the measurements to verify the robustness of the offset MPC controller, and the responses with and without sensor noise are compared. Specifically, zero mean standard deviation Gaussian noise of 0.1% is added to the thrust, LPC and HPC surge margins measurements. The simulation results of offset MPC with and without sensor noise for the system configuration with the battery-ultracapacitor are shown in Fig. 5.17.

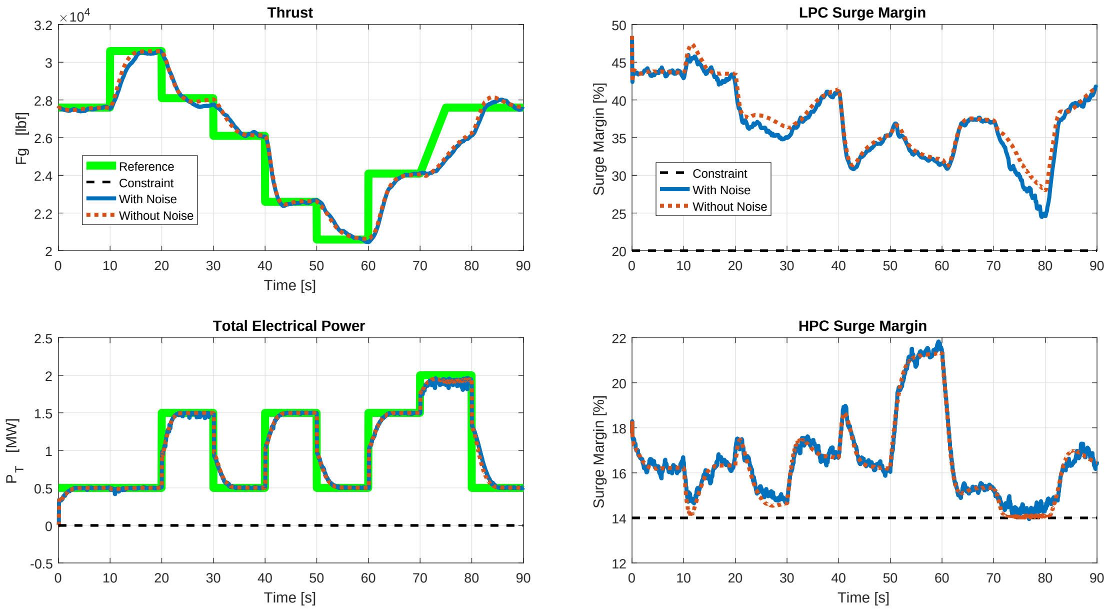

Figure 5.17: Comparison of offset MPC controllers with and without sensor noise.

The simulation results show that the offset MPC controller is able to handle the sensor noise. In the time interval between 70 to 80 sec, when high electrical power is required and thrust increment is requested near the HPC surge margin limit, some oscillations are observed because of the sensor noise, but the controller is able to handle the situation without having surge margin violations. Note that the computational delay can be accommodated by using advanced-step MPC [105] and other delays, if they exist, can be handled by augmenting the discrete-time model with extra delay states.

5.6 Conclusions

In this chapter, the development of a coordinated control strategy for a gas turbine engine, an advanced dual generator subsystem, and energy storage elements for MEA and AEA in the presence of large transient thrust and electrical loads has been pursued. The control design exploited rate-based MPC, for which various enhancements and design options have been considered and analyzed. Specifically, single MPC, MMPC, and offset MPC strategies were applied and compared to uncoordinated LQR and integrated LQR strategies in full nonlinear model simulations.

The comparison of closed-loop responses for the cases of uncoordinated LQR and integrated LQR indicated that the integrated control is capable of outperforming uncoordinated control in terms of tracking performance. The integrated LQR controller yielded better thrust and total electrical power tracking than the integrated MPC controller, however, with more surge margin constraint violations and without granting any protection against violation of other constraints.

To improve prediction model accuracy, MMPC and offset MPC design approaches were pursued. In the latter case, auxiliary offset states are used to represent the error between the linear model-based estimates of the constrained outputs and their actual value from the nonlinear model. The simulation results show that a single MPC with offset states is able to satisfy the surge margin constraints while MMPC, a more complex controller, has some constraint violations. Thus, the single rate-based offset MPC controller appears to be the best strategy to control the system. The closed-loop system performance with different types of energy storage elements has also been analyzed with the combined battery and the ultracapacitor pack providing the best solution; however, the weight and size impact of such an approach need to be carefully analyzed. Including energy storage elements into the system improves performance. For instance, in our simulations, the offset MPC controller with battery-ultracapacitor pack improved average thrust deviation by 6.4%, settling time of thrust by 3.47%, average total electrical power tracking by 29.45%, and settling time of total electrical power by 8.65% compared to an offset MPC controller applied to the system without energy storage, and without any constraint violations. In addition, the offset MPC controller was able to handle sensor noise. Our results support the perspective that the aircraft architecture with dual generators attached to different gas turbine engine shafts and battery-ultracapacitor pack, controlled by a single offset MPC controller, is appealing in terms of fast, safe and efficient thrust and electrical load delivery for future MEA and AEA.

CHAPTER VI

Conclusions

In this dissertation, we consider situations where resources are limited, other constraints are imposed on the system, and no knowledge or probabilities of the future are available. The goal is to develop/design methods of resource management for the available resources and for a constrained system operating in an UDE. Resources may include different notions, such as fuel, power, capabilities, energy, and so on. In this dissertation, resource management is divided into two main categories: 1) planning, and 2) control. At the planning level, the set of tasks to be performed is scheduled based on the limited resources, to maximize performance and resource use. At the control level, the system controller is designed to follow the schedule by considering all the system constraints for safe and efficient operations. For the best resource management performance in constrained dynamic situations, the planning level and the control level need to be considered together.

In Chapter 2, the RFSM is defined; it is a composed FSM with global and/or local input restrictions that takes into account resource limitations. Then, ReRFSM is defined, based on composition and pruning operations of the FSM that can handle UDEs. To obtain the resource management policy, DP for ReRFSM is developed, and to obtain a solution with faster computation time, LBFS for ReRFSM is developed. However, we show that the optimal policies of the ReRFSM are not achieved by combining the local optimal policies of the RFSMs in the ReRFSM because of UDE, where a random FSM can be added or deleted without preview information. In the case of ReFSM, DP for ReFSM may stay near the optimal policy because the local optimal policies are connected to each other by the union policy, but in the case of ReRFSM, the local optimal policies are not well connected because of input restrictions. Furthermore, DP for ReRFSM may be worse than LBFS for ReRFSM, so DP may not be useful in UDE. Obtaining the optimal policy of ReRFSM is impossible without knowing the future perfectly, so for real-world situations with UDE and limited resources, using LBFS for ReRFSM is reasonable.

In Chapter 3, a multi-function phased array radar task scheduler is designed using ReRFSM in UDE for planning level resource management. The scheduling algorithm is developed based on several different methods: DP for ReRFSM, SWC for ReRFSM, BFS for ReRFSM, and LBFS for ReRFSM. The simulation results indicate that the algorithm performs similarly for all methods, but LBFS for ReRFSM is computationally faster than the other methods, especially, when the number of threats is large. Therefore, using a LBFS for ReRFSM is effective in this application. The resource management results depend on a cost function that is based on a heuristic; the results are intuitive and acceptable for real-world scenarios. A distributed architecture using communication for fleet-level radar systems is also developed. The simulation results show that the distributed architecture performs better than the decentralized one by yielding better overall performance metrics and handling more threats in the same battle situation. The results indicate that cooperation is important to utilize the limited resource well, and our approach allows convenient design of cooperative resource management strategies.

In Chapter 4, patrolling mission planners for an UAV and multi-UAV are designed using ReRFSM in UDE for planning level resource management. By applying DP for ReRFSM, a policy is generated and by applying LBFS for ReRFSM, a sub-optimal solution is generated with faster computation time. LBFS for ReRFSM can handle around 400 waypoints in real time in the current setting, while DP for ReRFSM can handle approximately 10 waypoints; for large numbers of waypoints, LBFS for ReRFSM should to be used. 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. The simulation results show that the resulting schedule is affected by the priorities of the waypoints and no fly zones, as well as their changes, so they should be considered to obtain the plan for the UAV.

In Chapter 5, a rate-based MPC controller for a gas turbine engine and electrical power system for future MEA/AEA in the presence of large transient thrust and electrical loads is designed for control level resource management. In the presence of large transient thrust and electrical loads, the interactions between the subsystems are significant, so they have to be considered. To alleviate the effects of the interactions, a two-generator configuration is exploited and advanced energy storage elements are considered. The control design exploits rate-based MPC to handle a variety constraints, for which various enhancements and design options have been considered and analyzed. The comparison of closed-loop responses for the cases of uncoordinated and integrated designs indicates that the integrated control is better than uncoordinated control in terms of the performance metrics. Our results show that the two-generator configuration and battery-ultracapacitor pack, controlled by a single offset MPC controller, is appealing in terms of fast, safe and efficient thrust and electrical load delivery for future MEA and AEA. This indicates that for control level resource management, considering the interactions between the subsystems and integrated control for the subsystems are important to achieve good tracking of the schedule and safe operations for the system.

As indicated previously, for the best resource management performance in constrained dynamic situations, the planning level and the control level need to be con-

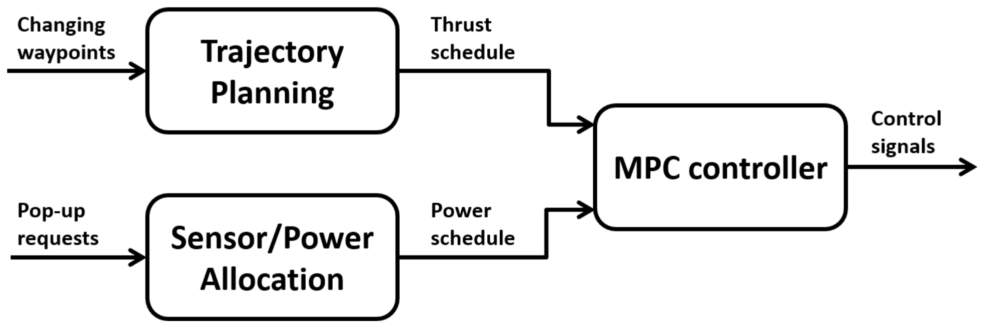

Figure 6.1: An example of joint operation of planning level and control level.

sidered together. For the planning level, cooperation is important to maximize use of the limited resource, and for the control level, considering the system constraints and interactions between the subsystems are important for good tracking of the schedule and safe operations of the system. Consider a two-aircraft patrolling mission in an UDE where each aircraft is equipped with two sensors for collecting information. Then, a patrolling mission planner similar to that from Chapter 4 can generate a trajectory plan by considering cooperation between the two aircraft, and generate corresponding thrust schedules for each aircraft. Also, a scheduler similar to that developed in Chapter 3 can provide the electrical power schedule for each aircraft's sensor by considering cooperation between the sensors, as well as the aircraft, to collect the information. Because of the UDE, the schedules can change in time without available prediction. For instance, the set of waypoints, and for each waypoint, the number of tasks may change in time without prediction. Once the schedules are obtained, a rate-based MPC controller for each aircraft, similar to that developed in Chapter 5, can provide control signals for each aircraft to follow their given schedule while considering the system constraints. The joint operation of the planning level and the control level of the given example is shown in Fig. 6.1. The planning level ensures good schedules, and the control level ensures good tracking of schedules and safe operations of the system, so good resource management performance in constrained dynamic situations can be achieved.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Datasheet k2 ultracapacutors - 2.85v/3400f. `http://www.maxwell.com/images/documents/K2_2_85V_DS_3000619EN_3_.pdf`. Accessed: 2016-06-20.

[2] Nanophosphate lithium ion prismatic pouch cell amp20m1hd-a. `http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm`. Accessed: 2016-06-20.

[3] Pratt & whitney's f135 advanced multi-variable control team receives utc's prestigious george mead award for outstanding engineering accomplishment, 2010.

[4] M. Alnajjar and D. Gerling. Control of three-source high voltage power network for more electric aircraft. In *International Symposium on Power Electronics, Electrical Drives, Automation and Motion*, pages 232–237, Jun 2014.

[5] N. Basilico, N. Gatti, and F. Amigoni. Developing a deterministic patrolling strategy for security agents. In *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies*, volume 2, pages 565–572, 2009.

[6] A. Behbahani, D. Culley, S. Garg, R. Millar, B. Smith, J. Wood, T. Mahoney, R. Quinn, S. Carpenter, B. Mailander, et al. Status, vision, and challenges of an intelligent distributed engine control architecture. *SAE Technical Paper 2007-01-3859*, 2007.

[7] A. R. Behbahani, A. Von Moll, R. Zeller, and J. Ordo. Aircraft integration challenges and opportunities for distributed intelligent control, power, thermal management, diagnostic and prognostic systems. *SAE Technical Paper 2014-01-2161*, 2014.

[8] G. Berry and L. Cosserat. The esterel synchronous programming language and its mathematical semantics. In *Seminar on Concurrency*, pages 389–448, Jul 1984.

[9] G. Berry, M. Kishinevsky, and S. Singh. System level design and verification using a synchronous language. In *Proceedings of the 2003 IEEE/ACM International Conference on Computer-aided Design*, pages 433–440, 2003.

[10] D. P. Bertsekas. *Dynamic Programming and Optimal Control Volume I*. Athena Scientific, 3rd edition, 2005.

[11] D. P. Bertsekas. *Dynamic Programming and Optimal Control Volume II.* Athena Scientific, 4th edition, 2012.

[12] D. Bertsimas and G. V. Ryzin. The dynamic traveling repairman problem. *Sloan School of Management, Massachusetts Institute of Technology*, 1989.

[13] W. Blair, G. A. Watson, T. Kirubarajan, and Y. Bar-shalom. Benchmark for radar allocation and tracking in ecm. *IEEE Transactions on Aerospace and Electronic System*, 34:1097–1114, Oct 1998.

[14] F. Bullo, E. Frazzoli, M. Pavone, K. Salva, and S. Smith. Dynamic vehicle routing for robotic systems. *Proceedings of the IEEE*, 99:1482–1504, 2011.

[15] G. L. Calhoun, M. H. Draper, M. F. Abernathy, M. Patzek, and F. Delgado. Synthetic vision system for improving unmanned aerial vehicle operator situation awareness. In *SPIE Enhanced and Synthetic Vision*, pages 219–230, May 2005.

[16] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Springer, 2nd edition, 2007.

[17] J. W. Chapman, T. M. Lavelle, J. S. Litt, and T.-H. Guo. A process for the creation of t-mats propulsion system models from npss data. In *Propulsion and Energy Forum*, Jul 2014.

[18] J. W. Chapman, T. M. Lavelle, J. S. Litt, R. D. May, and T.-H. Guo. Propulsion system simulation using the toolbox for the modeling and analysis of thermodynamic systems (t-mats). In *Propulsion and Energy Forum*, Jul 2014.

[19] N. Christofides. The vehicle routing problem. *Revue Francaise D'automatique, D'informatique et de Recherche operationnelle*, 10:55–70, 1976.

[20] B. Coltin, M. Veloso, and R. Ventura. Dynamic user task scheduling for mobile robots. In *AAAI Workshop on Automated Action Planning for Autonomous Mobile Robots*, Aug 2011.

[21] M. Corbett, P. Lamm, J. McNichols, M. Boyd, and M. Wolff. Effects of transient power extraction on an integrated hardware-in-the-loop aircraft/propulsion/power system. *SAE Technical Paper 2008-01-2926*, 2008.

[22] J. B. Cruz, J. M. A. Simaan, A. Gacic, H. Jiang, B. Letellier, M. Li, and Y. Liu. Modeling and control of military operations against adversarial control. In *Proceedings of the 39th IEEE Conference on Decision and Control*, volume 3, pages 2581–2586, Dec 2000.

[23] Q. Dang, I. Nielsen, K. Steger-Jensen, and O. Madsen. Scheduling a single mobile robot for part-feeding tasks of production lines. *Journal of Intelligent Manufacturing*, 25:1271–1287, 2013.

[24] G. Dantzig and J. Ramser. The truck dispatching problem. *Management Science*, 6:80–91, 1959.

[25] J. A. DeCastro. Rate-based model predictive control of turbofan engine clearance. *Journal of Propulsion and Power*, 23(4):804–813, 2007.

[26] J. C. DeLaat, R. D. Southwick, and G. W. Gallops. High stability engine control (histec). In *32nd Joint Propulsion Conference Cosponsored by AIAA, ASME and SAE*, Jul 1996.

[27] M. P. DeSimio, B. M. Hencey, and A. C. Parry. Online prognostics for fuel thermal management system. In *ASME 2015 Dynamic Systems and Control Conference*, page V001T08A003. American Society of Mechanical Engineers, Oct 2015.

[28] S. Di Cairano and H. Tseng. Driver-assist steering by active front steering and differential braking: Design, implementation and experimental evaluation of a switched model predictive control approach. In *IEEE Conference on Decision and Control*, pages 2886–2891, Dec 2010.

[29] C. Duron and J. Proth. Multifunction radar: Task scheduling. *Journal of Mathematical Modeling and Algorithm*, 1:105–116, Jun 2002.

[30] S. A. Edwards, N. Halbwachs, R. V. Hanxleden, and T. Stauner. Synchronous programming. In *Synchronous Programming*, Nov 2005.

[31] J. Enright, K. Salva, and E. Frazzoli. Stochastic and dynamic routing problems for multiple uninhabited aerial vehicles. *Journal of Guidance, Control, and Dynamics*, 32:1152–1166, 2009.

[32] S. S. Epp. *Discrete Mathematics with Applications.* Cengage Learning, 4th edition, 2010.

[33] M. Faied. Multistep classification problem using evsi bayesian preposterior framework. *Robotics and Autonomous Systems*, 72:277–284, 2015.

[34] M. Faied, I. Assanein, and A. Girard. Uavs dynamic mission management in adversarial environments. *International Journal of Aerospace Engineering*, pages 1–10, 2009.

[35] M. Faied and A. Girard. Modeling and optimization of military air operations. In *Proceedings of the 48th IEEE Conference on Decision and Control*, pages 6274–6279, Dec 2009.

[36] M. Faied, A. Mostafa, and A. Girard. Dynamic optimal control of multiple depot vehicle routing problem with metric temporal logic. In *IEEE American Control Conference*, Jun 2009.

[37] M. Faied, A. Mostafa, and A. Girard. Vehicle routing problem instances: Application to multi-uav mission planning. In *AIAA Guidance, Navigation, and Control Conference*, pages 565–572, Aug 2010.

[38] J. L. Fargeas, P. Kabamba, and A. Girard. Cooperative surveillance and pursuit using unmanned aerial vehicles and unattended ground sensors. *Sensors*, 15:1365–1388, 2015.

[39] J. Felder, H. Kim, and G. Brown. Turboelectric distributed propulsion engine cycle analysis for hybrid-wing-body aircraft. In *AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, Jan 2009.

[40] M. Figliozzi. Vehicle routing problem for emissions minimization. *Transportation Research Record: Journal of the Transportation Research Board*, 2:1–7, 2010.

[41] D. Ford. Planning for an unpredictable war: British intelligence assessments and the war against japan, 193745. *Journal of Strategic Studies*, 27(1):136–167, 2007.

[42] J. Fuller, I. Das, F. Potra, and J. Ji. System and method of applying interior point method for online model predictive control of gas turbine engines, Jun 2005. US Patent App. 11/150,703.

[43] E. Garone, S. Di Cairano, and I. Kolmanovsky. Reference and command governors for systems with constraints: A survey of their theory and application. *Automatica*, 75(1):306–328, 2017.

[44] S. Ghosh, J. Hansen, R. Rajkumar, and J. Lehoczky. Integrated resource management and scheduling with multi-resource constraints. In *Proceedings of the 25th IEEE International Real-Time Systems Symposium*, pages 12–22, Dec 2004.

[45] R. R. Hill, J. O. Miller, and G. A. McIntyre. Application of discrete event simulation to modeling military problems. In *Proceedings of the Winter Simulation Conference*, volume 1, pages 780–788, Feb 2001.

[46] M. Huang, H. Nakada, S. Polavarapu, R. Choroszucha, K. Butts, and I. Kolmanovsky. Towards combining nonlinear and predictive control of diesel engines. In *American Control Conference*, pages 2846–2853, Jun 2013.

[47] M. Huang, K. Zaseck, K. Butts, and I. Kolmanovsky. Rate-based model predictive controller for diesel engine air path: Design and experimental evaluation. *IEEE Transactions on Control Systems Technology*, 2016.

[48] L. C. Jaw and J. Mattingly. *Aircraft Engine Controls: Design, System Analysis, and Health Monitoring.* AIAA, 1st edition, 2010.

[49] M. I. Jiménez, L. D. Val, J. J. Villacorta, A. Izquierdo, and M. R. Mateos. Design of task scheduling process for a multifunction radar. *IET Radar, Sonar and Navigation*, 6:341–347, 2012.

[50] R. M. Karp and M. Held. Finite-state processes and dynamic programming. *SIAM Journal on Applied Mathematics*, 15(3):693–718, May 1967.

[51] H. D. Kim. Distributed propulsion vehicles. In *27th International Congress of the Aeronautical Sciences*, Sep 2010.

[52] J. S. Kim, K. M. Powell, and T. F. Edgar. Nonlinear model predictive control for a heavy-duty gas turbine power plant. In *American Control Conference*, pages 2952–2957, Jun 2013.

[53] I. Kolmanovsky, L. Jaw, W. Merrill, and H. T. Van. Robust control and limit protection in aircraft gas turbine engines. In *Proceedings of 2012 IEEE Multiconference on Systems and Control*, Oct 2012.

[54] I. Kolmanovsky and W. Merill. Limit protection in gas turbine engines based on reference and extended command governors. In *Proceedings of 50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, Jul 2014.

[55] M. Kvasnica, P. Grieder, and M. Baotić. Multi-parametric toolbox (mpt), 2004.

[56] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithm. *Eropean Journal of Operational Research*, 59:345–358, 1992.

[57] S. Lin and B. Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operation Research*, 21:498–516, 1973.

[58] J. S. Litt, D. L. Simon, S. Garg, T.-H. Guo, C. Mercer, R. Millar, A. Behbahani, A. Bajwa, and D. T. Jensen. A survey of intelligent control and health management technologies for aircraft propulsion systems. *Journal of Aerospace Computing, Information, and Communication*, 1(12):543–563, 2004.

[59] C. A. Luongo, P. J. Masson, T. Nam, D. Mavris, H. D. Kim, G. V. Brown, M. Waters, and D. Hall. Next generation more-electric aircraft: A potential application for hts superconductors. *IEEE Transactions on Applied Superconductivity*, 19:1055–1068, Jun 2009.

[60] N. A. Lynch and M. R. Tuttled. An introduction to input/output automata. *CWI Quarterly*, 2(3):219–246, 1989.

[61] F. Maraninchi and N. Halbwachs. Compositional semantics of non-deterministic synchronous languages. In *Programming Languages and System*, pages 235–249, Apr 1996.

[62] F. Maraninchi and Y. Remond. Argos: An automaton-basedsynchronous language. *Computer Languages*, 27:61–92, Apr 2001.

[63] J. S. McCarley and C. D. Wickens. Human factors implications of uavs in the national airspace. Technical Report AHFD-05-5/FAA-05-1, Aviation Human Factors Division, Savoy, IL, 2005.

[64] J. S. McGrew, J. P. How, L. A. Bush, B. Williams, and N. Roy. Air-combat strategy using approximate dynamic programming. *Journal of Guidance, Control, and Dynamics*, 33:1641–1654, 2010.

[65] C. McMillen and M. Veloso. Distributed, play-based role assignment for robot teams in dynamic environments. In *Distributed Autonomous Robotic Systems 7*, pages 1271–1287, 2006.

[66] I. R. McNab. Pulsed power for electric guns. *IEEE Transactions on Magnetics*, 33:453–460, Jan 1997.

[67] S. L. C. Miranda, C. J. Baker, K. Woodbridge, and H. Griffiths. Comparison of scheduling algorithms for multifunction radar. *IET Radar, Sonar and Navigation*, 1:414–424, 2007.

[68] S. L. C. Miranda, C. J. Baker, K. Woodbridge, and H. Griffiths. Fuzzy logic approach for prioritisation of radar tasks and sectors of surveillance in multifunction radar. *IET Radar, Sonar and Navigation*, 1:131–141, 2007.

[69] S. L. C. Miranda, C. J. Baker, K. Woodbridge, and H. D. Griffiths. Phased array radar resource management: A comparison of scheduling algorithms. In *Proceedings of the IEEE Radar Conference*, pages 79–84, 2004.

[70] A. J. Mitcham and J. J. A. Cullen. Permanent magnet generator options for the more electric aircraft. In *International Conference on Power Electronics, Machines and Drives*, pages 241–254, Jun 2002.

[71] I. Moir and A. Seabridge. *Aircraft Systems: Mechanical, Electrical and Avionics Subsystems Integration*. Wiley, 3rd edition, 2008.

[72] P. W. Moo. Scheduling for multifunction radar via two-slope benefit functions. *IET Radar, Sonar and Navigation*, 5:884–894, 2011.

[73] P. W. Moo and Z. Ding. Coordinated radar resource management for networked phased array radars. *IET Radar, Sonar and Navigation*, 9:1009–1020, 2015.

[74] K. Muehlbauer and D. Gerling. Two-generator-concepts for electric power generation in more electric aircraft engine. In *International Conference on Electrical Machines*, pages 1–5, Sep 2010.

[75] P. J. Norman, S. J. Galloway, G. M. Burt, J. E. Hill, and D. R. Trainer. Evaluation of the dynamic interactions between aircraft gas turbine engine and electrical system. In *IET Conference on Power Electronics, Machines and Drives*, pages 671–675, Apr 2008.

[76] P. Oberlin, S. Rathinam, and S. Darbha. Today's traveling salesman problem. *IEEE Robotics & Automation Magazine*, 17:70–77, 2010.

[77] P. Ortner and L. del Re. Predictive control of a diesel engine air path. *IEEE Transactions on Control Systems Technology*, 15:449–456, 2007.

[78] H. Psaraftis. Dynamic vehicle routing problem. *Vehicle Routing: Methods and Studies*, 1988.

[79] P. Rakhra, P. J. Norman, S. J. Galloway, and G. M. Burt. Modelling and simulation of a mea twin generator uav electrical power system. In *International Proceedings of Universities' Power Engineering Conference*, pages 1–5, Sep 2011.

[80] P. J. Ramadge and W. H. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25:206–230, 1987.

[81] J. Rawlings and D. Mayne. *Model Predictive Control: Theory and Design.* Nob Hill Publishing, 1st edition, 2009.

[82] H. Richter. *Advanced Control of Turbofan Engines.* Springer, 1st edition, 2012.

[83] J. A. Rosero, J. A. Ortega, E. Aldabas, and L. Romeral. Moving towards a more electric aircraft. *IEEE Aerospace and Electronic Systems Magazine*, 22:3–9, Mar 2007.

[84] K. Salva, F. Bullo, and E. Frazzoli. On travelling salesperson problem for dubins' vehicle: Stochastic and dynamic environment. In *Conference on Decision and Control, and the European Control Conference*, pages 4530–4535, Dec 2005.

[85] K. Salva, E. Frazzoli, and F. Bullo. On the point-to-point and travelling salesperson problem for dubins' vehicle. In *American Control Conference*, volume 2, pages 786–791, Jun 2005.

[86] U. S. A. F. C. Scientist. Technology horizons: A vision for air force science and technology 2010-30. Technical report, United States Air Force, 2010.

[87] J. Seok, I. Kolmanovsky, and A. Girard. Integrated/coordinated control of aircraft gas turbine engine and electrical power system: Towards large electrical load handling. In *IEEE Conference on Decision and Control*, pages 3183–3189, Dec 2016.

[88] J. Seok, J. Zhao, J. Selvakumar, E. Sanjaya, P. T. Kabamba, and A. Girard. Radar resource management: Dynamic programming and dynamic finite state machines. In *European Control Conference*, pages 4100–4105, Jul 2013.

[89] T. A. Severson and D. A. Paley. Distributed optimization for radar mission coordination. In *American Control Conference*, pages 5102–5107, Jun 2012.

[90] B. Shahsavari, M. Maasoumy, A. Sangiovanni-Vincentelli, and R. Horowitz. Stochastic model predictive control design for load management system of aircraft electrical power distribution. In *American Control Conference*, pages 3649–3655, Jul 2015.

[91] G. Shen and P. E. Caines. Hierarchically accelerated dynammic programming for finite-state machines. *IEEE Transaction on Automatic Control*, 47(2):271–283, Feb 2002.

[92] S. Smith, M. Pavone, F. Bullo, and E. Frazzoli. Dynamic traveling repairman with priority demands. In *IEEE Conference on Decision and Control*, pages 1206–1211, 2008.

[93] S. Smith, M. Pavone, F. Bullo, and E. Frazzoli. Dynamic vehicle routing with priority classes of stochastic demands. *SIAM Journal on Control and Optimization*, 48:3224–3245, 2010.

[94] E. Thirunavukarasu, R. Fang, J. A. Khan, and R. Dougal. Evaluation of gas turbine engine dynamic interaction with electrical and thermal system. In *Electric Ship Technologies Symposium*, pages 442–448, Apr 2013.

[95] P. Thunholm. Planning under time pressure: An attempt toward a prescriptive model of military tactical decision making. *H. Montgomery, R. Lipshitz & B. Brehmer (Eds.), How Experts Make Decisions*, 2005.

[96] C. Ting, H. Zishu, and T. Ting. Dwell scheduling algorithm for multifunction phased array radars based on the scheduling gain. *Journal of Systems Engineering and Electronics*, 19:479–485, June 2008.

[97] R. Todd, D. Wu, J. A. dos Santos Girio, M. Poucand, and A. J. Forsyth. Supercapacitor-based energy management for future aircraft systems. In *Applied Power Electronics Conference and Exposition*, pages 1306–1312, Feb 2010.

[98] J. Tumova, G. Hall, S. Karaman, E. Frazzoli, and D. Rus. Least-violating control strategy synthesis with safety rules. In *16th International Conference on Hybrid Systems: Computation and Control. ACM*, 2013.

[99] J. Tumová, L. I. Reyes-Castro, S. Karaman, E. Frazzoli, and D. Rus. Minimum-violating planning with conflicting specifications. In *American Control Conference*, pages 200–205, Jun 2013.

[100] A. Vahidi, I. Kolmanovsky, and A. Stefanopoulou. Constraint handling in a fuel cell system: A fast reference governor approach. *IEEE Transactions on Control Systems Technology*, 15(1):86–98, 2007.

[101] M. Veloso, J. Biswas, B. Coltin, S. Rosenthal, S. Brandao, T. Mericli, and R. Ventura. Symbiotic-autonomous service robots for user-requested task in a multi-floor building. In *IROS Cognitive Assistive Systems Workshop*, Oct 2012.

[102] N. Visnevski, V. Krishnamurthy, S. Haykin, B. Currie, F. Dilkes, and P. Lavoie. Multi-function radar emitter modelling: A stochastic discrete event system approach. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 6295–6300, Dec 2003.

[103] G. A. Watson and W. D. Blair. Revisit calculation and waveform control for a multifunction radar. In *Proceedings of the 32nd IEEE Conference on Decision and Control*, pages 456–461, Dec 1993.

[104] R. R. Weber. Optimization and control. University of Cambridge, 1999.

[105] V. M. Zavala and L. T. Biegler. The advanced-step nmpc controller: Optimality, stability and robustness. *Automatica*, 45:86–93, 2009.

[106] J. Zhao, J. Seok, J. Selvakumar, R. Bencatel, P. T. Kabamba, and A. Girard. A greedy policy for fleet-level radar resource management. In *Conference on Decision and Control*, pages 3160–3165, Dec 2013.

[107] A. Zinnecker, J. W. Chapman, T. M. Lavelle, and J. S. Litt. Development of a twin-spool turbofan engine simulation using the toolbox for modeling and analysis of thermodynamic systems (t-mats). In *Propulsion and Energy Forum*, Jul 2014.