A NOVEL DATA-DRIVEN DESIGN PARADIGM FOR AIRLINE DISRUPTION MANAGEMENT

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Kolawole E. Ogunsina

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Daniel DeLaurentis, Chair

School of Aeronautics and Astronautics

Dr. Karen Marais

School of Aeronautics and Astronautics

Dr. Ilias Bilionis

School of Mechanical Engineering

Dr. Dengfeng Sun

School of Aeronautics and Astronautics

Approved by:

Dr. Gregory Blaisdell

Head for the Graduate Program of Aeronautics and Astronautics

This dissertation is dedicated to Samuel Omowonuola, Ololade Rachael, Morayo Abisola, Modupe Arike and Lawrence Isola of the Ogunsina family from Nigeria.

ACKNOWLEDGMENTS

First, I acknowledge God for the knowledge of everything. Second, I thank my family and friends for their immeasurable support. Third, I thank Southwest Airlines for generously providing the data used for this project. Fourth, I thank Blair Reeves, Chien Yu Chen, Kevin Wiecek, Dave Harrington, Jeff Agold, Rick Dalton, and Phil Beck, at Southwest Airlines, for their expert inputs in abstracting the data used for this work. Next, I thank my major professor, Dr. DeLaurentis, for the opportunity to be a part of the CISA laboratory during my graduate school days at Purdue University. Finally, I thank everyone that contributed to the success of this project in some way or manner.

TABLE OF CONTENTS

Page

LIST OF TABLES . . . viii

LIST OF FIGURES . . . ix

ABBREVIATIONS . . . xii

ABSTRACT . . . xv

1 INTRODUCTION . . . 1
- 1.1 Airline Scheduling . . . 2
- 1.2 Airline Disruption Management . . . 3
- 1.3 Resolution Paradigms for Airline Disruption Management . . . 4
  - 1.3.1 Classification by System Paradigm . . . 5
  - 1.3.2 Classification by Operations Recovery Paradigm . . . 6
- 1.4 Motivation for a new Paradigm for Airline Disruption Management . . . 10
- 1.5 Research Statement . . . 14
- 1.6 Research Questions and Tasks . . . 14

2 NOVEL PARADIGM FOR AIRLINE DISRUPTION MANAGEMENT . . . 16
- 2.1 Simultaneously-integrated Recovery Design Paradigm for Airline Disruption Management . . . 16
- 2.2 Proposed *SIR* Approach for Airline Disruption Management . . . 18
- 2.3 Intelligent Managers for an Automated Decision-making Framework . . . 19
  - 2.3.1 Intelligent Resolution . . . 21
  - 2.3.2 Intelligent Learning . . . 23
- 2.4 Comparison between Existing and Proposed *SIR* Paradigms for Airline Disruption Management . . . 24

3 EXPLORATORY DATA ANALYSIS FOR A *SIR* PARADIGM IN AIRLINE DISRUPTION MANAGEMENT . . . 31
- 3.1 Data Overview . . . 32
- 3.2 Data Analysis . . . 38
  - 3.2.1 Feature Abstraction . . . 40
  - 3.2.2 Feature Transformation . . . 45
  - 3.2.3 Dimensionality Reduction . . . 51
  - 3.2.4 Feature Selection . . . 57

4 CREATING INTELLIGENT AGENTS FOR A *SIR* PARADIGM IN AIRLINE DISRUPTION MANAGEMENT . . . 68

Page
4.1 Data Preprocessing . . . 71
4.2 Uncertainty Transfer Function Model (UTFM) . . . 73
4.2.1 UTFM Methodology . . . 74
4.2.2 Solution Approach for UTFM . . . 79
4.2.3 Computational Setup and Analysis . . . 89
4.2.4 UTFM Input and Output Features . . . 91
4.2.5 UTFM Learning . . . 95
4.2.6 UTFM Decoding . . . 98
4.2.7 UTFM Results . . . 101
4.3 Predictive Transfer Function Model (PTFM) . . . 105
4.3.1 PTFM Methodology . . . 107
4.3.2 Solution Approach for PTFM . . . 114
4.3.3 Single Phase ANN Development . . . 116
4.3.4 Multi-Phase ANN Inference . . . 122
4.3.5 Computational Setup . . . 125
4.3.6 PTFM Analysis and Results . . . 126

5 ENABLING INTEGRATION AND INTERACTION OF INTELLIGENT AGENTS FOR A *SIR* PARADIGM IN AIRLINE DISRUPTION MANAGEMENT . . . 136
5.1 A Symbiotic Synthesis of AI and DLT . . . 139
5.1.1 Artificial Intelligence . . . 139
5.1.2 Distributed Ledger Technology and Blockchain . . . 140
5.1.3 Decentralized AI for Airline Disruption Management . . . 141
5.2 A Case Study . . . 151
5.2.1 AI Calibration . . . 153
5.2.2 DLT Protocol . . . 159
5.2.3 Results and Discussion . . . 170

6 OUTRO . . . 178
6.1 Summary . . . 179
6.1.1 A Novel Paradigm for ADM . . . 179
6.1.2 Exploratory Data Analysis for a *SIR* Paradigm in ADM . . . 180
6.1.3 Creating Intelligent Agents for a *SIR* Paradigm in ADM . . . 181
6.1.4 Enabling Integration and Interaction for a *SIR* Paradigm in ADM 182
6.2 Recommendations for Future Work . . . 184

REFERENCES . . . 185

A Nomenclature for Chapters 3, 4 & 5 . . . 198
A.1 Nomenclature for determinate aleatoric data features . . . 198
A.2 Nomenclature for indeterminate aleatoric data features . . . 199
A.3 Nomenclature for epistemic data features . . . 203

B Algorithms for Chapters 4 & 5 . . . 206

Page
B.1 Dynamic Programming Algorithm 1 206
B.2 Dynamic Programming Algorithm 2 207
B.3 Dynamic Programming Algorithm 3 208
B.4 Dynamic Programming Algorithm 4 209

C Single-Layer Neural Network as Gaussian Process 210

D Testing Methods for Algorithms and Processes 211
D.1 Static Testing 211
D.2 Dynamic Testing 212
D.2.1 Data Flow Testing 212
D.2.2 Random Testing 214
D.2.3 Coverage Criterion Testing 214

VITA 216

LIST OF TABLES

Table Page

1.1 Limitations to current paradigm for airline operations recovery 13

2.1 Properties of *SIR* paradigms for airline disruption management 25

3.1 Disruption outlook for functional domains in Southwest Airlines network operations control from September 2016 to September 2017 34

3.2 Lengthscales of refined data features for predicting actual turnaround duration through Gaussian process regression 64

4.1 Data preprocessing for development of intelligent agents for disruption management 72

4.2 List of features for *Intra-State* HMMs in UTFM. 90

4.3 List of features for *Inter-State* HMMs in UTFM. 91

4.4 Epistemic features for multi-phase ANN inference in PTFM 125

4.5 Model performance summary of ANNs for all functional roles in AOCC . 129

4.6 Input features and corresponding values for a specific disrupted DAL-HOU flight. 132

4.7 Target features and corresponding values for a specific disrupted DAL-HOU flight. 133

5.1 Pseudocount and inherent likelihood of observing new values for all functional roles in Southwest Airlines network operations control 158

5.2 Position weights of states at separate phases of activity in the UTFM for an intelligent agent 168

5.3 Total number of queued disrupted flight schedules for each functional role in SWA-NOC used for case study 171

5.4 Summaries of the first ten disruption resolution events from Hashgraph data structure shown in Fig. 5.4 174

LIST OF FIGURES

Figure Page

1.1 The airline scheduling process 1

1.2 The disruption management process [4] 3

1.3 Resolution paradigms for airline disruption management 9

1.4 Current (sequential) practice for airline disruption management [9] 10

2.1 Automatic System for Airline Disruption Management 18

2.2 Flowchart for IROPS domain manager in intelligent Multi-Agent System . 20

2.3 Verification and validation process for Multi-Agent Systems [46] 29

3.1 Data analysis procedure for a *SIR* design paradigm 39

3.2 Sample knowledge abstraction of a basic flight schedule for airline disruption management 41

3.3 Feature transformation process of raw data for airline disruption management 47

3.4 Principal component analysis of indeterminate aleatoric features for Weather domain 53

3.5 t-Distributed Stochastic Neighborhood Embedding analysis of indeterminate aleatoric features for Weather domain 56

3.6 Mutual information of flight schedule data features for Weather Domain with respect to actual turnaround duration 60

3.7 Probability densities of test turnaround duration and mean predictions of turnaround duration for delayed flight schedule instances in Weather domain 62

3.8 Mean predicted turnaround duration data vs. test turnaround duration data 63

3.9 Quantile-Quantile plot of standard mean error between predicted and test data for actual turnaround duration 65

4.1 Disruption management for a scheduled flight defined by a Markov decision process 75

4.2 RDBN architecture for a representative flight leg 78

4.3 Component assembly approach for automatic uncertainty quantification for disruption management 80

Figure Page

4.4 Probabilistic graphical model representation of UTFM 81

4.5 Intra-state HMM schema for remaining in an activity phase in UTFM . . . 85

4.6 Inter-state HMM schema for transitioning between activity phases in UTFM86

4.7 Phases of disruption management with respect to schedule execution . . . 95

4.8 State transition graph of optimal *Intra-State* HMM for remaining in turnaround decision (TAD) 98

4.9 State transition graph of optimal *Inter-State* HMM for transition from turnaround decision (TAD) to turnaround outcome (TAO) 100

4.10 Probabilistic graphical map for UTFM assessment of a specific disrupted DAL-HOU flight 102

4.11 Probabilistic graphical map for UTFM assessment of a specific disrupted MDW-BOS flight 104

4.12 Timeline horizon during flight schedule execution 107

4.13 Fundamental model of a single neuron in an artificial neural network . . 110

4.14 Artificial neural network framework for estimating target flight schedule features 111

4.15 Flowchart of general process for ANN learning 113

4.16 Component assembly approach for automatic estimation of performance measures for disruption management 115

4.17 Flowchart of parallel ensemble approach for PTFM estimation 123

4.18 ANN predictions vs. test data for strategic disruption management . . . 127

4.19 ANN predictions vs. test data for tactical disruption management 128

4.20 ROC plots of ANN classifiers for operational disruption management . . 130

5.1 Human involvement in disruption management and operations recovery at the AOCC 136

5.2 Taxonomy of a decentralized AI platform for airline disruption management143

5.3 Integration and interaction routine in a decentralized AI platform for airline disruption management 151

5.4 Hashgraph data structure revealing a recovery plan from a few functional roles in SWA-NOC 172

Figure Page

5.5 Elapsed time period before first round of consensus for an increasing number of functional roles in SWA-NOC 176

D.1 Software testing routine for *i-MAS* algorithms and processes 211

ABBREVIATIONS

AI	Artificial Intelligence
ABFT	Asynchronous Byzantine Fault Tolerance
ADM	Airline Disruption Management
ANN	Artificial Neural Network
AOCC	Airline Operations Control Center
AR	Aircraft Recovery
ASAS	Automatic or Semi-Automatic System
ATC	Air Traffic Control
AUC	Area Under Receiver Operating Characteristic Curve
BCE	Binary Cross Entropy
BFT	Byzantine Fault Tolerance
CDM	Collaborative Decision Making
CR	Crew Recovery
DAG	Directed Acyclic Graph
DBN	Dynamic Bayesian Network
DBQS	Database Query System
DLT	Distributed Ledger Technology
DSS	Decision Support System
ED	Enroute Decision
ES	Enroute Schedule
EO	Enroute Outcome
FAA	Federal Aviation Administration
FN	False Negative
FP	False Positive
GPR	Gaussian Process Regression

GQN	Generic Q-Negotiation
HMM	Hidden Markov Model
IATA	International Air Transport Association
ICE	Information Cross Entropy
i-MAS	Intelligent Multi-Agent System
IoT	Internet of Things
IR	Integrated Recovery
IROPS	Irregular Operations
MALG	Models and Algorithms
MAS	Multi-Agent System
MASDIMA	Multi-Agent System for Disruption Management
MIR	Mutual Information Regression
OR	Operations Research
PCA	Principal Component Analysis
PGM	Probabilistic Grapical Model
PIR	Partial Integrated Recovery
PTFM	Predictive Transfer Function Model
QQ	Quantile-Quantile
RDBN	Relational Dynamic Bayesian Network
RMSE	Root Mean Square Error
ROC	Receiver Operating Characteristic
SIR	Simultaneously-Integrated Recovery
SWA-NOC	Southwest Airlines Network Operations Control
TAD	Turnaround Decision
TAS	Turnaround Schedule
TAO	Turnaround Outcome
TID	Taxi-In Decision
TIS	Taxi-In Schedule
TIO	Taxi-In Outcome

TN	True Negative
TOD	Taxi-Out Decision
TOS	Taxi-Out Schedule
TOO	Taxi-Out Outcome
TP	True Positive
TSA	Transportation Security Administration
t-SNE	t-distributed Stochastic Neighborhood Embedding
UTFM	Uncertainty Transfer Function Model

ABSTRACT

Ogunsina, Kolawole E. Ph.D., Purdue University, December 2020. A Novel Data-Driven Design Paradigm for Airline Disruption Management. Major Professor: Daniel DeLaurentis.

Airline disruption management traditionally seeks to address three problem dimensions: aircraft scheduling, crew scheduling, and passenger scheduling, in that order. However, current efforts have, at most, only addressed the first two problem dimensions concurrently and do not account for the propagative effects that uncertain scheduling outcomes in one dimension can have on another dimension. Uncertainties in scheduling outcomes originate from random disruption events (like inclement weather and aircraft malfunction), the order in which the events occur, and how they are resolved. As such, these uncertainties propagate through all problem dimensions for airline disruption management on the day of operation.

In addition, existing approaches for airline disruption management include human specialists who decide on necessary corrective actions for airline schedule disruptions on the day of operation. However, human specialists are limited in their ability to process copious amounts of information imperative for making robust decisions that simultaneously address all problem dimensions during disruption management. Therefore, there is a need to augment the decision-making capabilities of a human specialist with quantitative and qualitative tools that can rationalize complex interactions amongst all dimensions in airline disruption management, and provide objective insights to the specialists in the Airline Operations Control Center (AOCC). To that effect, we provide a discussion and demonstration of an agnostic and systematic paradigm for enabling simultaneously-integrated recovery of all problem dimensions during airline disruption management, through an intelligent multi-agent system that employs principles from artificial intelligence and distributed ledger technology.

1. INTRODUCTION

Irregular operations (IROPS) constitute one of the most challenging problems faced by the airline industry. Random disruptive events such as inclement weather and equipment (aircraft) malfunction can negatively impact airline operations more so than they would on any other mode of transportation [1]. Most airlines are often forced to delay (or cancel) flights in order to resume scheduled operations. Disruptions induced by irregular operations on scheduled airline operations result in displaced flight crews and passenger delays, yielding an increased total annual operating cost of about three to five percent of the airline revenue [1]. Hence, efforts to mitigate the effects of irregular operations on scheduled day-to-day flight operations of an airline are planned and monitored by the Airline Operations Control Center (AOCC) to maintain the overall efficiency of airline operations. The AOCC consists of specialized human teams with specific (local) objectives (for example, the aircraft team is primarily responsible for having the right aircraft on a disrupted flight), which are subordinate to the overall airline (global) objective of minimizing the impacts of deviations from the planned airline schedule (i.e. maximizing recovery of scheduled operations) [2].

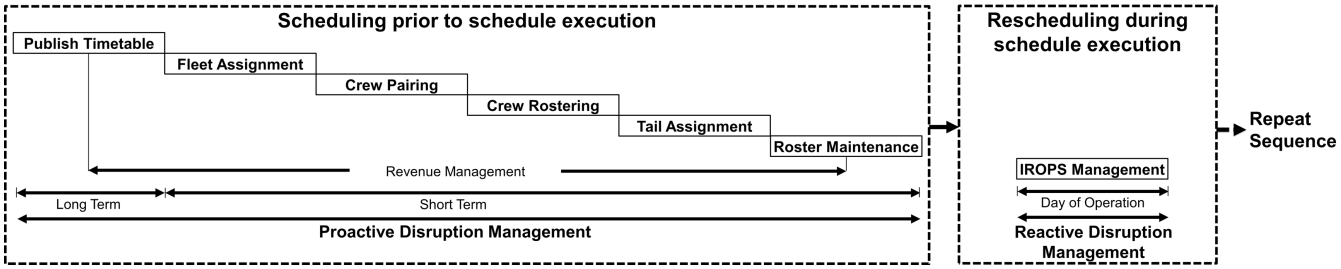

Figure 1.1. The airline scheduling process

1.1 Airline Scheduling

Understanding airline disruption management requires a brief introduction of the overarching problem of airline scheduling. Airlines try to maximize profit (or minimize loss) by solving problems that arise during the scheduling process shown in Fig. 1.1. The scheduling process represents a paramount long-term and short-term planning mechanism of every airline, wherein resources (i.e. aircraft and crew) available to an airline are paired with a certain amount of passenger demand for air travel [3] that effectively define three interdependent problem dimensions - aircraft, crew, and passenger [4]. A typical airline schedule is the principal outcome of the airline scheduling process that reveals the flights offered to customers on a particular day of operation. This schedule includes assigned aircraft types, departure and arrival airports, and time of day details on how the operation of each flight unfolds from turnaround at the departure airport to aircraft gate-parking at the destination airport.

Specifically, the airline scheduling process can be defined by two separate and periodic phases with respect to schedule execution. The first phase, which is airline scheduling prior to schedule execution, commences with the publishing of a timetable of flight services that will be offered to passengers over a certain time period (usually between six to nine months). Next, revenue management commences such that the goal is to maximize revenue by selling tickets at certain prices to different types of customers. The allocation of airline resources (such as aircraft and crew) are scheduled during revenue management, at which point fleet assignment is used to designate aircraft fleet types to different flights in order to determine the number of available seats in each flight. Upon completing the fleet assignment, the airline begins the crew scheduling process, where the first step is to define necessary crew duty periods (i.e. crew pairing) that will accommodate for all flights in the timetable over a specific period of time, usually one month. After obtaining the crew pairings, crew members are designated to the pairings (i.e. crew rostering) and an individual personalized crew roster is made available to crew members. As flight dates draw closer, each

aircraft in the fleet is assigned to a specific flight in the tail assignment step [3], and the aircraft-crew pair roster is continually revised to facilitate changes that may occur during the second phase of the airline scheduling process (i.e. rescheduling during schedule execution on the day of operation) [5, 6].

1.2 Airline Disruption Management

Airline disruption management is a subprocess of the airline scheduling process. Upon effectively attaining an optimal schedule through proactive disruption management during airline scheduling prior to execution, the airline is left with monitoring the execution of the schedule, by means of the AOCC, on the day of operation. A disruption is defined as a state during the execution of an otherwise optimal schedule, where deviation from the schedule is sufficiently large that it has has to be substantially changed [7]. The purpose of developing optimal flight schedules is defeated if disruptions abound later on during operation and cause significant deviations from the original schedule. Hence, airline disruption management is the process of solving problems related to aircraft, crew and passengers when a significant deviation from the optimal schedule obtained prior to execution occurs during schedule execution on the day of operation. In that regard, reactive disruption management during

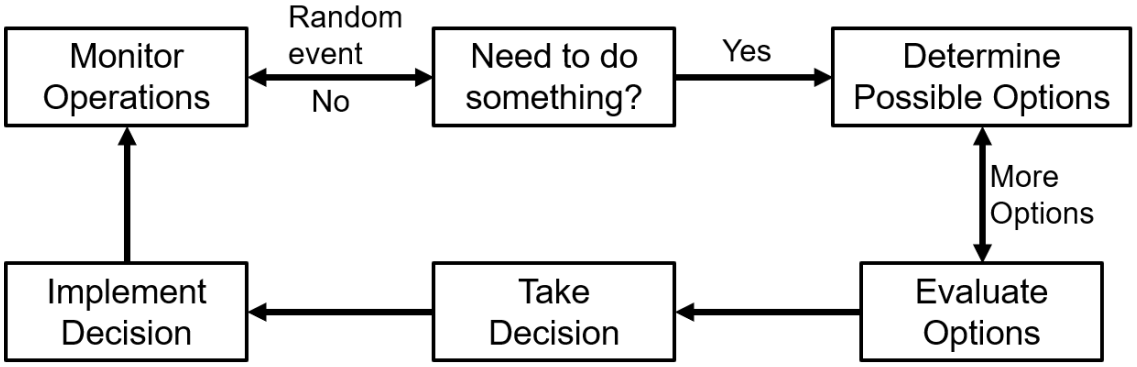

Figure 1.2. The disruption management process [4]

schedule execution typically begins when airline scheduling for proactive disruption management prior to schedule execution ends. Fig. 1.2 shows a generic disruption management process used in industry today to mitigate changes in planned schedule and improve IROPS efficacy on day of operation.

1. *Monitor Operations*: This is the first step in disruption management where flights, crew check-in, and passenger check-in are monitored to verify that the planned airline schedule for the day is still valid.

2. *Need to do something?*: If an unexpected event occurs such as inclement weather or aircraft malfunction, the event is evaluated to see whether or not further action is required. If not, monitoring of operations continues. If a corrective action is required, then one or more problem dimensions needs to be resolved.

3. *Identify and Evaluate possible options*: Using available information, the AOCC is tasked with finding and evaluating candidate resolutions to issues affecting each problem dimension in the airline schedule. The AOCC generally addresses a disruption by sequentially resolving issues related to aircraft fleet, crew members, and passengers respectively in that order, primarily through the use of *rules-of-thumb* (i.e. aircraft swap, flight rerouting, crew swap, etc.) amassed from the experience of human specialists in each problem dimension [8].

4. *Take Decision*: A decision is taken based on the list of plausible resolutions.

5. *Implement Decision*: The decision is implemented and the airline schedule is updated accordingly.

1.3 Resolution Paradigms for Airline Disruption Management

[8] and [9] proposed two different classification schemes to describe the system design and recovery methods that are being, or can be, used by the AOCC to address disruptions in airline operations. [9] analyzed sixty compelling research works in airline operations recovery, published between 1984 and 2012. The schemes categorize

different resolution paradigms used for airline disruption management by the type of systems that can be used to recover operations, and the problem dimensions that can be solved during disruptions.

1.3.1 Classification by System Paradigm

[9] developed a system paradigm classification to include both commercially available software and tools only available in an academic setting for airline disruption management, and are as follows:

1. *Models and Algorithms (MALG)*: This class of system paradigms refers to research work that proposes mathematical models and algorithms to resolve any or all of the problems domains in the AOCC.

2. *Database Query Systems (DBQS)*: This is the most commonly used class of system paradigms at airlines, which allows human specialists in the AOCC to query existing databases in order to obtain necessary data for decision-making. While DBQS may be fairly easy to develop and implement, they are limited by the fact that the best decision is solely dependent on the human specialist. The quality of the resolution to a particular disruption is dependent on the knowledge and experience of the human operator, thus making DBQS susceptible to the propagation of epistemic uncertainty during disruption management.

3. *Decision Support Systems (DSS)*: This class of system paradigms for airline disruption management has the same properties as the DBQS, but with additional and enhanced capabilities to support decision-making of human specialists in the AOCC. Unlike DBQS, DSS are able to analyze large volumes of data and present disruption resolutions that consider more variables in order to enable better and informed decision-making by AOCC specialists.

4. *Automatic or Semi-Automatic Systems (ASAS)*: This class of system paradigms for airline disruption management automate repetitive tasks in the resolution

generation and evaluation steps of the disruption management process, thereby effectively replacing the functional part of the AOCC with computerized programs. In a completely automatic system, all the decision-making is performed by the system while some decisions are made by human controllers in a semi-automatic system.

Analysis of the classification of system design paradigms for airline operations recovery, performed by [9], reveal that 75% of the sixty research works reviewed fall under *Models and Algorithms (MALG)*, and as such, are not included in current tools or systems used by the AOCC. About 22% of the research works were classified as *Decision Support Systems (DSS)*, which indicates that they can be incorporated in tools used by human specialists in the AOCC for disruption management, while 3% of the works reviewed classified as *Automatic or Semi-Automatic Systems (ASAS)*.

1.3.2 Classification by Operations Recovery Paradigm

[8] and [9] expressed in their respective works that resolution paradigms can also be classified based upon the problem dimension in which a resolution is applied. These classifications are as follows:

1. *Aircraft Recovery (AR)*: This class of resolution paradigms for operations recovery only addresses the aircraft domain. A lot of research work in aircraft recovery utilize operations research (OR) methods, such as network flow models, graph theory algorithms, time-band optimization, etc, to minimize total passenger delays on an airline network when one or more scheduled aircraft are out of service. For example, [10] used a mixed integer multi-commodity flow model, coupled with Dantzig Wolfe decomposition [11], to address disruptions due to aircraft swaps. The model proved to yield high quality solutions when tested using data from a Swedish domestic airline. Another popular method adopted for solving the aircraft recovery problem is meta heuristics. [12] combined Greedy Random Adaptive Search Procedure (GRASP) [13] and Ant Colony Optimiza-

tion (ACO) [14] to create an hybrid heuristic procedure that minimizes the total cost of reassignments of aircraft to flights, and delaying and cancellation of flights.

2. *Crew Recovery (CR)*: This class of resolution paradigms for operations recovery solves only the problems in the crew domain. Similar to aircraft recovery, most literature on crew recovery use conventional OR methods or hybrid methods that combine OR methods with search heuristics. However, unlike resolution paradigms for aircraft recovery which are mainly dependent on the type of method applied, resolution paradigms for crew recovery hinge on the assumptions made by researchers. Most solution paradigms assume a fixed flight schedule during crew recovery wherein aircraft recovery is assumed to be first complete. For instance, [15] adopt column generation to develop and solve an optimization model that integrates crew scheduling and crew pairing for a fixed flight schedule, such that columns are created by determining shortest paths through a depth-first search strategy. Other solution approaches do not assume or require a fixed flight schedule and allow for flight cancellations and delays. [16] use a rolling approach to obtain an efficient crew recovery schedule, where a sequence of optimization assignment problems is solved such that flights are recovered in order of increasing departure times.

3. *Partial Integrated Recovery (PIR)*: Partial integrated recovery paradigms address at least two of the three main problem dimensions in airline operations recovery. Research for these resolution paradigms have just recently emerged over the last decade and have significantly less literature than aircraft recovery and crew recovery. Similar to aircraft recovery and crew recovery, paradigms for partial integrated recovery adopt optimization and/or heuristics to minimize airline operating costs together with some measure of passenger disruption costs. [17] studied an approach to integrate fleet assignment with crew scheduling during airline planning, such that the Benders decomposition method [18]

for generating feasibility cuts from aircraft maintenance routing is combined with a crew duty flow model. The authors note that a major challenge encountered during the development of the multi-commodity network flow model was addressing the cascading effect caused by disruptions from both problem dimensions. Complementary to the work by [17] which addressed the aircraft and crew domains in airline disruption management, [19] proposed a dual-optimization model that simultaneously developed recovery plans for aircraft and passenger domains, such that airline operating costs and estimated passenger delay and disruption costs are minimized to determine which flight leg departures to postpone and which ones to cancel.

4. *Integrated Recovery (IR)*: This class of resolution paradigms is able to recover all three problem (aircraft, crew, passenger) domains in operations recovery separately but not concurrently. Analysis performed by [9] reveal that there are only a handful of literature on this class of disruption resolution paradigms (three proposals between 1996 and 2013), due to the increased complexity of the problem and computational time when a monolithic process is employed to obtain an integrated solution. The authors believe that a non-monolithic method, such as a distributed system design supported by an acceptable decision-making mechanism, can reduce the complexity of the integrated problem. [20] were the first to present computational results on a fully integrated airline operations recovery problem. The authors use a backtracking optimization approach, via a Benders decomposition scheme, to develop and solve a schedule recovery model. The schedule recovery model is the master problem that relates several variables from different sub-problems that represent the problem dimensions in airline operations recovery. However, a sub-problem resolution order is inherently imposed by the optimization algorithm, thereby making some problem dimensions more important than others. This approach was tested using data from a major US airline with a dense network, and showed to be effective when no more than 65% of the flights in the network are subject to disruption. [20].

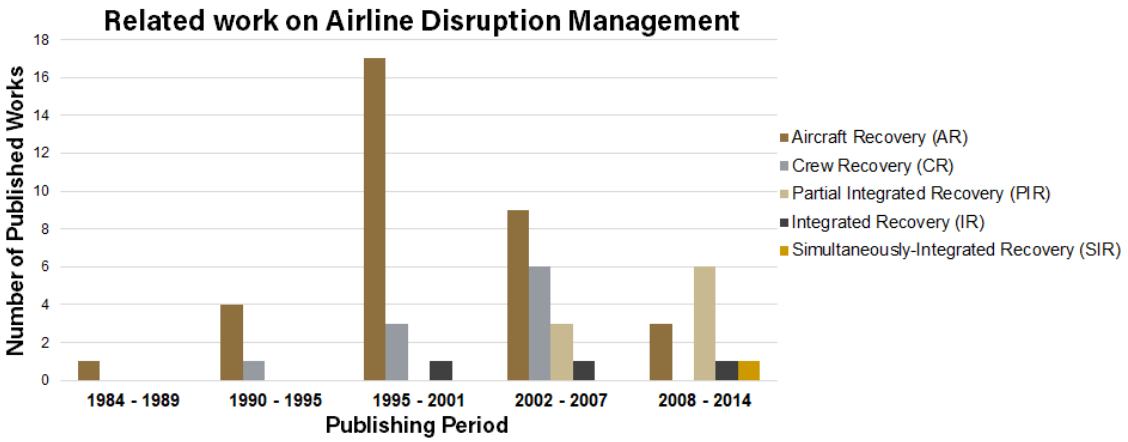

Figure 1.3. Resolution paradigms for airline disruption management

5. *Simultaneously-Integrated Recovery (SIR)*: This class of resolution paradigms is able to recover all problem dimensions in airline operations simultaneously, such that the hierarchy (or importance) of one problem dimension over another is eliminated. There has been only one approach, by [9], that uses this resolution paradigm till date. The authors use a multi-agent system design paradigm to characterize the AOCC, such that human roles in each problem dimension - with the most frequent tasks - are performed by intelligent agents. The approach uses an adaptive protocol, called the Generic-Q-Negotiation (GQN) [21], to ensure multi-attribute negotiation, with several rounds of feedback, between two types of agents (organizer and respondent) in order to achieve consensus. However, this prototype disruption resolution approach and most of those discussed in the operations recovery paradigms aforementioned, are unable to handle evolving information dynamics (i.e. uncertainty in information evolution) during the disruption management process.

Fig. 1.3 shows the evolution of the amount of literature on the different classes of operations recovery paradigms. There is significantly less literature for *IR* and *SIR* when compared to other recovery paradigms over the three-decade period, such that

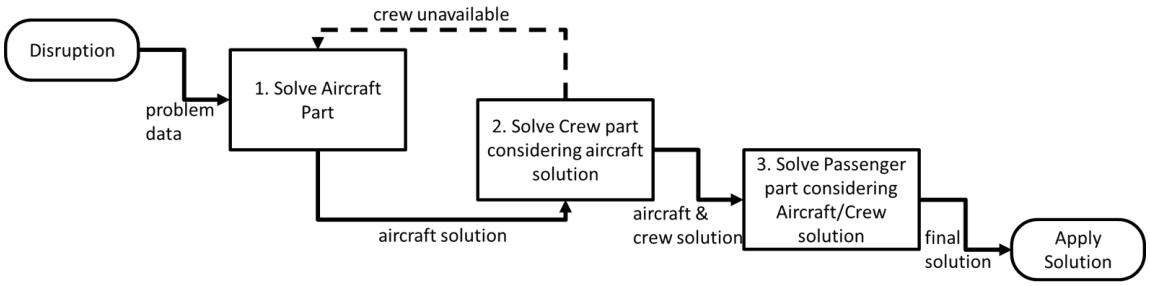

Figure 1.4. Current (sequential) practice for airline disruption management [9]

the very few available literature for these classes of paradigms recently emerged over the last decade.

1.4 Motivation for a new Paradigm for Airline Disruption Management

As evidenced in the previous section, current resolution approaches for addressing the airline operations recovery problem, during disruption management, have generally focused on using traditional operations research (i.e. optimization and heuristics) methods that resolve one or more (but not all) of the primary problem dimensions (aircraft schedule recovery, crew schedule recovery, and passenger schedule recovery). These schedule recovery approaches try to achieve one or more of a set of possible objectives, such as: minimizing the cost of reserve crews and spare aircraft used; minimizing passenger recovery costs; minimizing loss of passenger goodwill; and minimizing the amount of time until it is possible to resume the original schedule [22]. Regardless of the objective, the airline operations recovery problem must be solved within minutes, and this time limitation makes it unrealistic to solve large detailed optimization models, as such, to meet these objectives, most airline recovery processes are sequential [23].

Fig. 1.4 shows the current practice for airline disruption management. When a disruption in scheduled operations occurs, the AOCC typically reacts by resolving the problem in a sequential manner where issues related to the aircraft fleet, crew

members, and passengers are addressed respectively, in that order, by their corresponding human specialists. These specialists, stationed in the AOCC, proactively monitor and mitigate problems and disruptions related to aircraft, crew members, and passengers in the airline network. The resolutions implemented by human specialists in each phase of the airline recovery process influence the resolutions applied in subsequent phases, and hence, the global objectives of the AOCC are achieved such that the overall airline recovery is restricted to a specific order during resolution (i.e. aircraft-crew-passenger).

According to the Amadeus IT group (a major IT group for the global travel industry), one of the main drivers that has significantly contributed to the lack of progress in developing a full solution to airline disruption management is limited bandwidth of human specialists [1]. Several key decisions at each phase of the recovery practice shown in Fig. 1.4, such as corrective actions implemented for a certain disruption type, are made by human managers in the AOCC. Although human managers are flexible in decision-making, they are not capable of parsing the copious amounts of data necessary for simultaneously making robust real-time decisions for all the problem dimensions in operations recovery. In major airlines, adding more personnel to the AOCC does not effectively increase human bandwidth because of the significantly large size of the airline network [1].

Furthermore, most decision support systems used to solve each problem dimension in the current recovery practice, are often deterministic and require the assessment of a human specialist before the resolutions generated by these support systems, at one or more phases of the recovery, are implemented as corrective action [8]. In addition to aleatoric uncertainty stemming from the random occurrence of disruptive events (like bad weather) on the day of operation, the current practice introduces epistemic uncertainty at each phase of the recovery when human specialists, with different levels of experience, are required to make decisions that will affect the solution generated in the subsequent phase.

Moreover, almost all of the resolution paradigms currently in practice adopt traditional monolithic system design methods where specifications are first developed before a system that meets the specifications is subsequently designed [1]. For instance, robust airline scheduling, a fairly new and integrated recovery paradigm for airline operations, aims to augment existing airline planning models to include both the costs associated with performing the original airline schedule and the expected costs of recovering the schedule from disruptions [2]. However, adding supplemental features to the existing planning model increases its complexity, and consequently increases the computational time needed to effectively recover a disrupted airline schedule. In addition, many existing decision support systems are unable to simultaneously address all the problem dimensions in airline operations, while recovering the original airline schedule, partly due to the propagation and evolution of disruptions during the operations recovery process.

Finally and most importantly, majority of the operations research methods adopted in current decision support systems do not explicitly account for the passengers' view of the journey during disruption management [24]. What may be considered as a minor delay from the AOCC's perspective (with respect to the reallocation of airline resources during a disruption) may be a significant disruption to a business executive that misses a crucial meeting or a family whose long-awaited vacation is delayed. In 2012, over 150 million people were affected by airline disruptions in the United States, of which 17 million experienced missed connections and cancelled flights [25]. According to Ira Gershkoff, the Principal Consultant at Travel Technology Research Ltd: "There is every reason to believe the historic challenge of re-routing planes, crew and passengers during disruption will finally be addressed over the next several years. As system and procedural innovations improve technical capabilities, the people side will evolve as well, making better decisions and executing coordinated responses." [26] Although the expected revenue lost from loss of passenger goodwill from airline disruption may not be readily quantifiable via the current sequential recovery practice, proper characterization of different disruption types specifically re-

Table 1.1. Limitations to current paradigm for airline operations recovery

Drawbacks of Current Practice for Airline Disruption Management
1. *Sequential Resolution Approach*: Solving aircraft, crew, then passenger problems in that order imposes an inherent constraint on the design (and solution) space for airline operations recovery. 2. *Deterministic Resolutions*: Resolutions obtained from most decision support tools in practice do not account for uncertainties in decision-making by human specialists and evolving information at each stage of flight on day of operation. 3. *Monolithic System Design*: Specifications continuously evolve as new capabilities are developed for existing recovery methods, thereby rapidly increasing system complexity and thus making monolithic system design less desirable for resolution platforms for airline operations recovery.

lated to (or caused by) passengers (e.g. delays due to security check-in) can serve as a proxy for estimating and minimizing loss of passenger goodwill by effectively affording airlines the option to implement disruption resolutions that prioritize the re-accommodation of passengers over the reallocation of airline resources. To that effect, there is a need for a simple, heterogeneous, and integrated airline operations recovery paradigm that leverages data on different *rules-of-thumb* employed by human specialists in the AOCC to accurately and concurrently prescribe corrective solutions for all problem dimensions in the event of a disruption in original airline schedule, while addressing the limitations (highlighted in Table 1.1) to the current operations recovery practice.

1.5 Research Statement

The main objective of this research is to develop and implement a semi-automatic learning and decision-making paradigm that simultaneously recovers all problem dimensions during airline disruption management via a special type of consensus mechanism design, where the disruption resolutions obtained are unrestricted by the order in which the problem dimensions are solved. The *rules-of-thumb* used by human specialists in the AOCC are inculcated in an automated and non-monolithic decision support framework by employing data-driven methods like predictive analytics and machine learning techniques, such that decisions made to resolve a certain type of disruption influence the behaviors and decisions that are used to address the same (or similar) disruption in the future. This ensures that the decision-making framework learns from previous experiences in order to make better and informed decisions just like a human specialist would.

1.6 Research Questions and Tasks

In order to achieve the aforementioned research objectives, a total of five tasks will be completed to answer three research questions:

- *Can we use the statistics of predictive analytics and machine learning to characterize and assess the functional parts of the AOCC for intelligent decision support systems?*

    1. Develop uncertainty models by employing unsupervised machine learning to properly characterize evolutionary and propagative behavior for different types of disruption, which impact the resolution reliability for all problem dimensions during airline operations recovery and disruption management.
    2. Develop predictive models by means of supervised machine learning techniques, to prescribe the recovery impact of disruption resolutions across

all problem dimensions at each phase of flight schedule execution during airline disruption management.

- *Can we achieve simultaneous decision-making among intelligent systems for all problem dimensions in airline operations recovery by leveraging recent improvements in consensus mechanism design?*

  1. Develop hashgraph protocol for achieving consensus among disruption resolutions prescribed by intelligent systems for all problem dimensions during airline disruption management.
  2. Integrate and test intelligent systems (i.e. uncertainty and predictive models), for operations recovery and disruption management across all problem dimensions with consensus protocol.

- *How can we assess the new airline operations recovery paradigm?*

  1. Identify and develop consistent metrics to measure performance and effectiveness of resolutions (or corrective actions) prescribed by intelligent systems for concurrent operations recovery.

2. NOVEL PARADIGM FOR AIRLINE DISRUPTION MANAGEMENT

This chapter introduces a novel simultaneously-integrated recovery (*SIR*) design paradigm for airline disruption management, and then discusses the intelligent domain manager; a key component of the paradigm that characterizes the decision-making process used by human managers in the AOCC for all problem dimensions in operations recovery. Lastly, we discuss the comparison between our new *SIR* design paradigm and the existing *SIR* approach for airline disruption management.

2.1 Simultaneously-integrated Recovery Design Paradigm for Airline Disruption Management

The simultaneously-integrated recovery paradigm is a design philosophy that enables the modeling of all support functions in the AOCC as a system of intelligent agents, such that their interactions are concurrently driven by local and global objectives. As such, this approach enables the realization of an acknowledged, distributed and scalable AOCC that can significantly reduce the problems encountered by existing AOCC organizations during disruption management. The acknowledged attribute of a new AOCC that adopts the *SIR* design paradigm ensures that while different teams (or intelligent agents) are responsible for their respective local objectives, their actions are subordinate to improving the global objective of the AOCC [27]. The distributed attribute is of three forms namely functional distribution, spatial distribution, and physical distribution [9]. Functional distribution allows the existing roles and functions in the AOCC to be distributed and managed using intelligent software agents. Spatial distribution allows the data and information utilized by different roles or intelligent agents to be distributed. For instance, data in different databases and

data in the same database but different partitions are spatially distributed. As such this attribute can help alleviate system and data integration problems in the AOCC. Physical distribution ensures that the roles and functions designated to different intelligent agents can be distributed to different machines, such that the agents have access to more computational resources. Lastly, the scalability attribute allows the AOCC framework to grow as long as functional and spatial requirements are satisfied accordingly. Based upon the combined effect from these attributes, the AOCC is best represented as a multi-agent system (MAS) [28, 29], such that support functions with frequently executed or repetitive tasks are performed by intelligent software agents. Human supervisors in the AOCC can also be modeled as software agents that make a final decision based upon candidate disruption resolutions. Some other benefits of adopting a MAS framework for enabling simultaneously-integrated recovery in the AOCC include:

- Ability to consider all problem dimensions (aircraft, crew, and passenger) at the same level of importance when generating resolutions.
- Increased autonomy and automation.
- Ability to measure local performance of constituent intelligent agents and global performance of the multi-agent system.
- Consideration of local preferences of each team (i.e. intelligent agent) responsible for solving different problem dimensions.
- Ability to consider environment dynamics based upon the fact that existing information can change while resolutions are being generated.
- Ability to generate solutions in real or almost-real time.

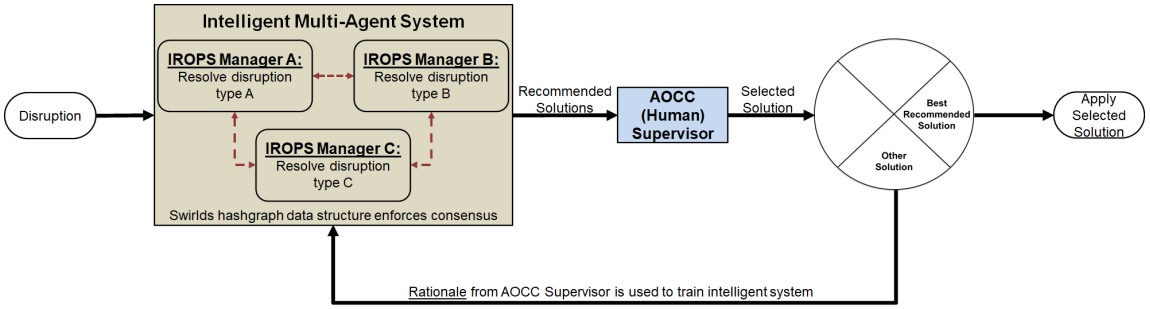

Figure 2.1. Automatic System for Airline Disruption Management

2.2 Proposed *SIR* Approach for Airline Disruption Management

Historically, majority of the research on decision support systems for the current recovery practice, shown in Fig. 1.4, primarily adopt general operations research (OR) methods [2] (i.e. integer optimization, network flow models, column generation methods, etc.), while the rest rely on meta heuristics to provide decision support to human specialists. Furthermore, only a few of the systems developed using these OR methods are included in tools that are used by human specialists in the AOCC today, and even much fewer are able to automate most of the functional (decision-making) parts of the AOCC by using computerized programs.

However, recent advancements in machine learning techniques, big data analysis, and mechanism design [30–32], coupled with cost-effective computational and data storage platforms [33], have presented an avenue for the development and evaluation of a new design paradigm for airline disruption management that addresses the antecedent drawbacks in the current recovery practice. Fig. 2.1 reveals an original automatic (or semi-automatic) system for airline disruption management wherein functional parts of the AOCC necessary for concurrent decision-making across all problem dimensions are characterized by an automated and intelligent multi-agent system (*i-MAS*) framework that predominantly employs predictive analytics and mechanism design. The light brown box in Fig. 2.1 defines the automated property of the overall approach where there is a simultaneous interaction among different intelligent special-

ist agents that represent separate managers for aircraft, crew and passenger domains. Each intelligent specialist agent is described by a data-driven and predictive decision-making model such as an artificial neural network. In the event of a disruption to the original airline schedule, each manager first predicts a set of disruption resolution actions -which a human specialist in its problem dimension would implement- that is independent of the resolution actions predicted by managers (specialist agents) from other problem dimensions. Next, the resolution actions by the manager are compared with predicted resolutions from other managers to ensure that their combined corrective actions do not result in a schedule conflict across all problem dimensions. A negotiation protocol in form of a consensus mechanism, such as hashgraph [32], can be invoked by a smart contract [34] to insure that disruption resolutions from each domain manager align with those from other managers so as to eliminate schedule conflicts during operations recovery and disruption management.

The final solution from the automated framework which contains the consensus (i.e. no schedule conflict) set of plausible corrective actions from all domain managers is then presented to a human AOCC Supervisor - shown in the light blue box in Fig. 2.1 - for approbation. If the best recommended action from the automated *i-MAS* framework is not implemented by the AOCC Supervisor then a rationale provided by the Supervisor and appropriated to the set of recommended corrective actions from the automated framework, is used to train the automated framework (and hence its learning property) for better-informed disruption resolutions in the future. The next section describes, in detail, the framework and mechanisms that the intelligent systems (i.e. IROPS managers) in the automated framework use for decision-making.

2.3 Intelligent Managers for an Automated Decision-making Framework

Before comparing the proposed *SIR* paradigm with the existing *SIR* paradigm for airline disruption management, it is necessary to first introduce the processes that

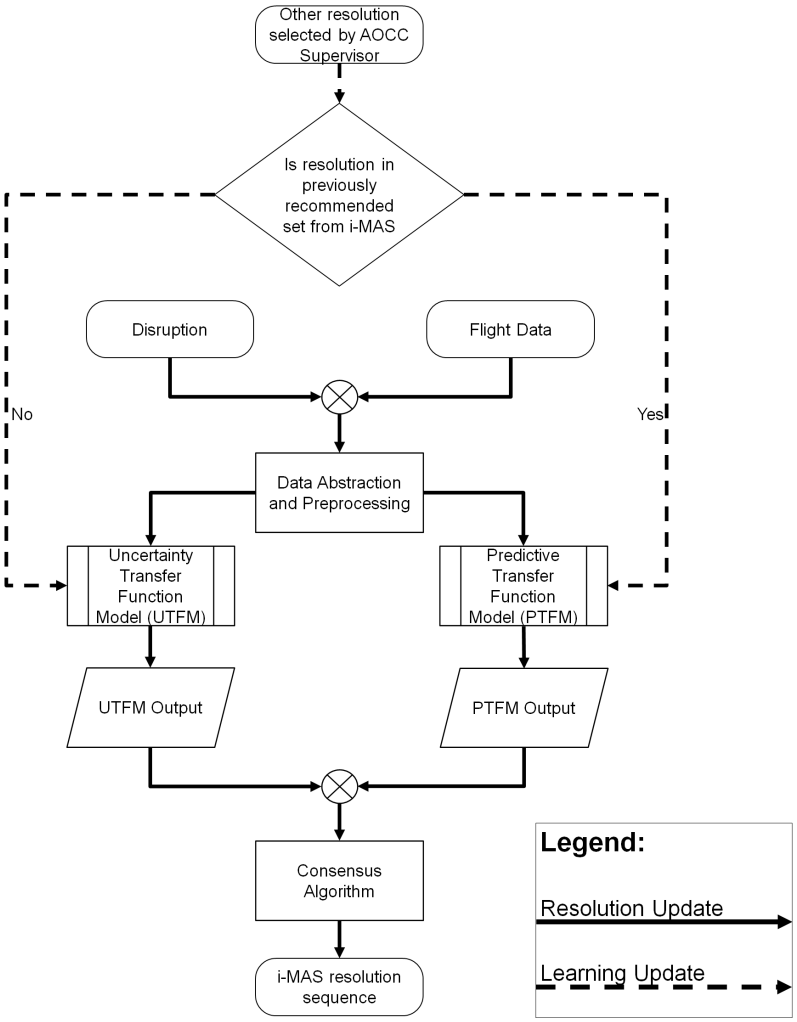

Figure 2.2. Flowchart for IROPS domain manager in intelligent Multi-Agent System

define the decision-making of IROPS domain managers for all problem dimensions in airline disruption management. Two separate timelines define the disruption resolution process of a domain manager in the automated *i-MAS* framework and are described by *Resolution Update* and *Learning Update*, as represented by solid and dashed arrows respectively in Fig. 2.2.

2.3.1 Intelligent Resolution

The *Resolution Update* timeline describes the manner in which a characteristic domain manager develops plausible resolutions to a certain type of disruption in the original airline schedule. The first step in the flow along the *Resolution Update* timeline is to obtain scheduled flight data and information on the corresponding type of disruption related to the problem dimension. Next, disruption information and flight schedule data are combined and transformed for use in machine learning algorithms (to enable predictive analytics) by employing data abstraction and preprocessing techniques [35].

Following data refinement, the uncertainty and predictive transfer function models (i.e. UTFM and PTFM) for the domain manager, represented in Fig. 2.2, are developed and managed by using historical data retrieved from primary air transportation stakeholders such as airlines. This data contains information on different types and instances of disruptions and their corresponding impact on original flight schedules and operations over a certain time period. The UTFM yields a set of plausible futures of decision actions for the most likely deviations from the original schedule based upon the long-term impact of uncertainty in schedule execution. In complement, the PTFM predicts short-term operations recovery metrics that represent the effectiveness and performance of the decision actions implemented by a domain manager during irregular operations. The predictions from the UTFM and PTFM (i.e. corrective actions) are subsequently ratified by ensuring that they align with the corrective actions from other IROPS domain managers, via a consensus algorithm as shown in Fig. 2.2, before they are presented to the AOCC Supervisor.

The transition from the first step (i.e. obtain disrupted flight information) to the last step (i.e. approbate disruption resolutions) along the *Resolution Update* timeline should be completed in minutes, in order to satisfy fast solution requirements by airlines and other air transportation stakeholders during disruption management. To that effect, we adopt a consensus framework that is based upon distributed ledger

technology (DLT) [36–38] to enable swift intelligent resolutions for disruption management. The components of the framework are described as follows:

- **Smart Contracts** are computerized transaction protocols that automatically execute the terms of a contract. They manage their internal states by using the state machine model, which allows for ease of framework development and programming [38]. Since smart contracts are self-enforcing upon satisfaction of contractual conditions, they minimize the need for trusted intermediaries thus making them very suitable for facilitating the development of a decentralized disruption resolution framework that mitigates malicious and accidental exceptions in decision-making. As such, smart contracts are useful for airline disruption management because they eliminate the need for intermediate human assessments of corrective actions generated by domain managers in an automated framework, which consequently eliminates the propagation of epistemic uncertainty during operations recovery. Smart contracts are executed on a distributed consensus mechanism design platform, such as hashgraph (a special type of blockchain-inspired technology), that defines the properties of the protocols executed by smart contracts.

- **Hashgraph** is a type of consensus mechanism design that uses gossip-about-gossip and virtual voting techniques to achieve fast, fair, and secure consensus thereby ensuring high fidelity negotiations amongst domain managers. The gossip-about-gossip and virtual voting properties of Hashgraph allows a domain manager to know what its disruption resolution will be and what that of other managers would be during a negotiation round. This allows all domain managers to achieve consensus on both the timing and the order in which disruption resolutions are made without having to make a formal decision during negotiation, which significantly accelerates the negotiation process. Hashgraph provides an ordering to an otherwise unordered set of disruption resolutions proferred by each manager after every consensus round (i.e. corrective actions

defined by resolutions from all domain managers) such that all domain managers receive an identical list of corrective actions as output. This property promotes the discovery, tracking, and validation of plausible corrective actions (disruption resolutions) that do not follow the sequential resolution paradigm currently used in industry today. Hashgraph also achieves asynchronous byzantine fault tolerance (ABFT) [39] by ensuring that consensus will be achieved with a probability of 1 if an IROPS domain manager influences (i.e. gossips) less than one-third of the disruption resolutions predicted by other managers during each round of negotiations, thus minimizing the number and increasing the overall quality of corrective actions recommended to the AOCC Supervisor by the intelligent multi-agent system framework.

2.3.2 Intelligent Learning

The *Learning Update* timeline, illustrated by the dashed arrows in Fig. 2.2, describes the manner in which the domain manager learns from the corrective actions implemented by the AOCC Supervisor. In a scenario where a different disruption resolution -other than the best-recommended resolution from any domain manager in the automated framework- is implemented by the human AOCC Supervisor in Fig. 2.1, the implemented resolution is used to train the affected domain manager in order to improve its decision-making process along the *Resolution Update* timeline in the future. If the disruption resolution implemented by the AOCC Supervisor is not among the list of previously recommended resolutions (i.e. new disruption resolution) from a domain manager, then the uncertainty model transfer function is updated. If the implemented resolution is among the corrective actions recommended by a domain manager, but is not the best-recommended corrective action as determined by the domain manager, then the predictive model transfer function is updated. Updates to the uncertainty and predictive transfer function models are periodic, so as to

improve the overall accuracy of disruption resolutions recommended by the domain managers over time.

The next section of this chapter discusses the similarities and differences between the extant *SIR* approach and the new *SIR* paradigm for airline disruption management, based upon the background provided in this section.

2.4 Comparison between Existing and Proposed *SIR* Paradigms for Airline Disruption Management

Simultaneously-integrated recovery (*SIR*) is the most recent class of approach for disruption management in airline operations recovery, which aims to recover all problem dimensions concurrently in a cooperative manner that does not grant priority to the order in which each problem dimension is resolved. The first and only existing published work on *SIR* was used to facilitate the implementation of a system called *MASDIMA - Multi-Agent System for Disruption Management*, which is based upon real data from TAP Air Portugal [9]. As such, we employ the principles and taxonomy of agent-based modeling [40,41] and multi-agent systems [42] to present the knowledge gap in *SIR* for airline disruption management, based upon an assessment of pertinent properties that define the *MASDIMA* and *i-MAS* frameworks. Table 2.1 shows a list of six fundamental properties that are necessary to classify and compare the existing and new multi-agent system SIR approaches for airline disruption management.

1. ***Agent-based Architecting Method***: This property defines the characterization of the rationale that informs the modeling of the internal architecture of human behavior in a representative multi-agent system for simultaneously-integrated recovery [41]. The existing (*MASDIMA*) *SIR* approach adopts a mathematical framework for defining human behavior, such that expressions such as the Boltzmann exploration and Q-value formulas [21] are used to describe the responsiveness of a domain manager, in order to represent the manner and threshold (i.e. probability) for which a human specialist would select a spe-

Table 2.1. Properties of *SIR* paradigms for airline disruption management

Paradigm Properties	**Existing *MASDIMA* Architecture**	**Proposed *i-MAS* Architecture**
Agent-based Architecting Method	Mathematical	Cognitive
Agent's Learning Environment	Model-free	Model-based
Computing	Distributed	Distributed
Decision Mechanism	Utility-based	Goal-based
Data Utilization (Statistical Efficiency)	Low	High
Verification and Validation	Structural	Structural

cific disruption resolution action over another during operations recovery. In contrast, our proposed *i-MAS SIR* approach adopts a cognitive framework that aligns closely with routine AOCC operations for defining human behavior. For instance, time-of-day events (e.g. departure and arrival times) are definable by a periodic vector, based upon the percentage of 8-hr work shift completed (at the time of departure or arrival) by human specialists in the AOCC. The work shift characterization can capture and represent daily disruption resolution proclivities of human specialists, which can be induced by how much time the specialists have to address a disruption before their work shift is complete.

2. ***Agent's Environment Learning Class***: This property describes the framework and manner in which intelligent agents (i.e. domain managers) in a *SIR* design paradigm learn to predict and optimize the consequence of their behavior in their respective environments for which different actions lead them from one state or situation to the next [42]. As shown in Table 2.1, the *MASDIMA* architecture adopts a model-free environment, wherein domain managers interact through reinforcement learning that does not use (nor estimate) a predefined

flight operations model consistent with airline scheduling practices to obtain optimal disruption resolutions during disruption management. As a result, the *SIR* implementation adopted in *MASDIMA* requires considerable trial-and-error experience to obtain acceptable estimates of future consequences from applying specific disruption resolutions. Conversely, our proposed *SIR* paradigm uses experience in form of historical data on airline scheduling and operations recovery to construct an internal model of transitions and immediate outcomes during different phases of flight to define a model-based environment for the multi-agent system. The model-based nature of the proposed *SIR* approach provides a statistically efficient way to use experience, such that every iota of information acquired from the modeling environment can be stored in a manner that is both statistically reliable and computationally flexible [43]. This enables constant replaning by domain managers and allows predicted disruption resolutions to be readily adaptive to evolving uncertainty (i.e. changing transition probabilities) at different phases of flight operations.

3. ***Computing***: This property describes the manner in which the domain managers of a multi-agent *SIR* design paradigm coordinate and communicate their individual actions to each another to achieve optimal and effective disruption resolutions. Both *SIR* approaches in Table 2.1 employ distributed computing [44], wherein intelligent agents address airline disruption management by independently solving different parts of the problem dimensions before communicating with each other through messages to agree on a unified solution.

4. ***Decision Mechanism***: The decision mechanism in a multi-agent system framework for simultaneously-integrated recovery during airline disruption management is the property that defines necessary methods and protocols for attaining unanimity by all intelligent agents (domain managers) on specific disruption resolutions. While both *SIR* design paradigms enable an automatic or semi-automatic system, the existing *SIR* approach in *MASDIMA* uses an

automated negotiation protocol called Generic Q-Negotiation (GQN) [21] as its decision mechanism. GQN provides a model-free reinforcement learning and adaptive protocol that negotiates multiple attributes and utilities (such as costs) among agents through several rounds of qualitative feedback across interdependent problem dimensions, thereby offering domain managers full or partial knowledge of each other's payoff to achieve consensus [9]. Contrary to GQN, which uses a utility-based decision mechanism to achieve consensus, our proposed *SIR* approach adopts a dynamic directed acyclic graph platform called hashgraph to enable autonomous time-sequencing of predicted disruption resolutions by intelligent domain managers, based upon substantive proof that each domain manager will efficiently predict the likelihood for employing its most probable sequence of corrective actions during operations recovery. To that effect, our proposed *SIR* design paradigm imposes a goal-based property on the *i-MAS* framework, wherein the goal of each intelligent domain manager in the semi-automatic system is to estimate its stake (i.e. probability of most likely sequence of corrective actions) before its predicted disruption resolution is considered when the swirlds hashgraph consensus algorithm [45] is invoked. Every domain manager in the proposed *SIR* design paradigm can achieve Byzantine agreement on any number of disruption resolutions, without voting (feedback) on disruption resolutions thereby allowing for zero bandwidth usage by the automated system, except for only communicating the hashgraph data structure. This significantly reduces the time needed to reach consensus, which is an advantage over the GQN protocol used in the existing *SIR* approach.

5. ***Data Utilization***: This property defines how much data is appropriated for developing multi-agent systems in a simultaneously-integrated recovery approach for airline disruption management. *MASDIMA's* model-free *SIR* approach provides an easy-to-use platform for functional decision-making interaction among intelligent agents but cannot leverage big data directly to improve real-time performance, because information from the modeling environment is

combined with dated and potentially inaccurate estimates or beliefs about state values representing different phases of flight operation. To that end, the existing *SIR* paradigm is statically inefficient. Moreover, information is stored in scalar quantities from which specific knowledge about rewards and transitions from moving from one flight phase to another cannot be untangled. As a result, the existing *SIR* approach in *MASDIMA* cannot readily accommodate for changes in contingencies and outcome utilities at different phases of flight during disruption management. Unlike the existing *SIR* approach, the our proposed *SIR* approach adopts a model-based method for developing a multi-agent system that is inherently statistically efficient and gets more accurate and amenable as more data becomes available [42].

6. ***Verification and Validation***: This property defines the method a *SIR* approach for a multi-agent system employs for determining its ability to produce valid and robust models that can serve as basis for decision makers in the AOCC. The existing and proposed *SIR* approaches for *MASDIMA* and *i-MAS* respectively adopt a structural validation method [46], as shown in Table 2.1. Fig 2.3 shows the general procedure for executing structural validation of multi-agent systems. The procedure commences with face validation, which ensures that animated behaviors and output trends are commensurate to real world trends. Next, significant parameters that affect behaviors and outputs are determined through sensitivity analysis, before the parameters are calibrated to fit real data by employing relevant optimization methods. Lastly, the results predicted from the multi-agent system are are sanctioned to ensure they match reality through output validation. Different results, parameters and processes are validated in both *SIR* paradigms. For the model-free *MASDIMA* architecture, equations and parameters used in the Q-Negotiation protocol that describe agents' architecting methods are verified through sensitivity analysis, while cost and delay estimates from different variations of the Q-Negotiation protocol are validated as a function of airline revenue. In our proposed model-based *SIR* paradigm,

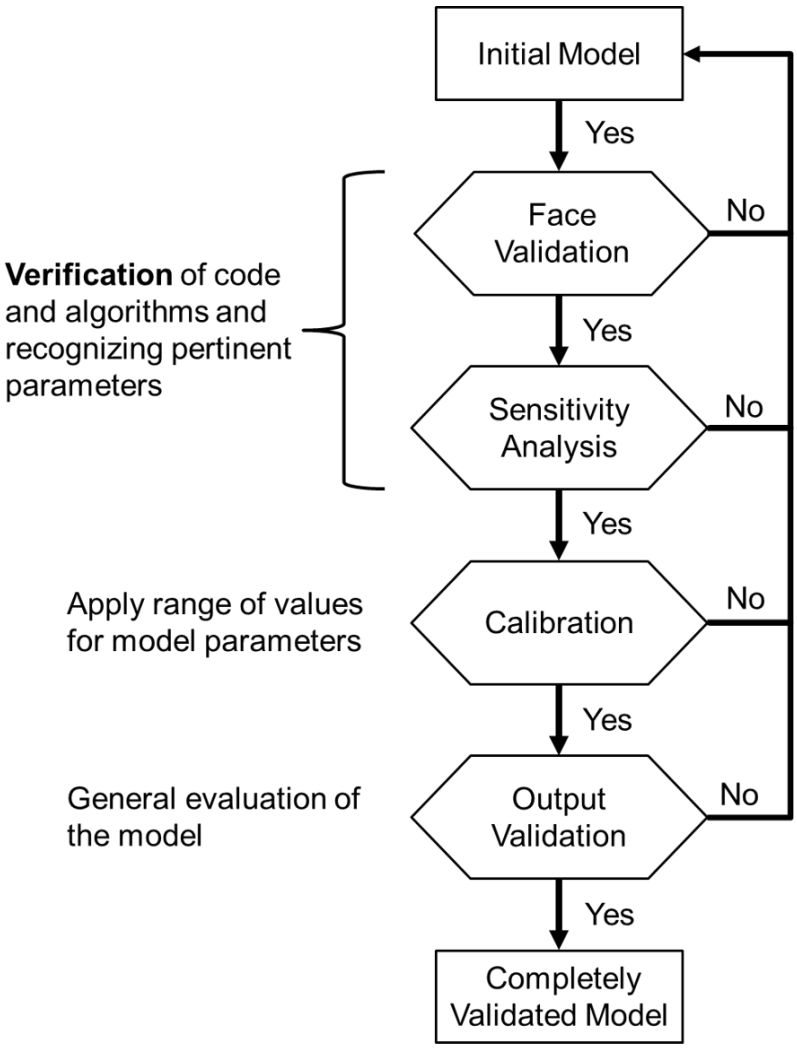

Figure 2.3. Verification and validation process for Multi-Agent Systems [46]

the accuracy and sensitivity of models representing different phases of flight operations are verified through statistics, while the order and quality of predicted disruption resolutions obtained by using the consensus algorithm are validated by estimating and comparing disruptions resolutions from real-world test data.

Chapter Summary: This chapter provided a high-level overview of our proposed *SIR* design paradigm for an automated and intelligent multi-agent system architecture, and its applicability for effectively improving airline disruption management. The chapter is concluded with a discussion of the comparison of crucial properties

that characterize our proposed *SIR* design paradigm and the only existing implementation of *SIR* for airline disruption management. The next chapter discuses, in extensive detail, the processes and techniques employed to analyze and prepare the historical data that inform the development and integration of constituent models that make up the *i-MAS* framework for achieving simultaneously-integrated recovery.

3. EXPLORATORY DATA ANALYSIS FOR A *SIR* PARADIGM IN AIRLINE DISRUPTION MANAGEMENT

Despite overcoming numerous financial and technical challenges over the last century through continued drive towards innovation and productivity, a complete solution to irregular operations in the airline industry has remained elusive. A major driver that has significantly stunted the progress in developing a full solution for airline disruption management is poor data integration and integrity.

Internal data containing real-time information about an airline's resources and its scheduled utilization over time, and external data for factors such as current and future weather forecasts, competitor activities, and air traffic control are necessary for efficient operations [24]. These bits of information and data must be readily available and accessible to represent drivers and constraints for scenarios induced by irregular operations, so as to facilitate the development of effective self-governing platforms for airline disruption management. In addition, whenever a new airline system is replaced or upgraded, new data sources are typically integrated into the existing framework [26]. The new data must be maintained for both existing and new applications, and thus present cost-intensive challenges for mitigating disruption because many facets of the airline infrastructure are impacted.

While there have been consistent improvements to the existing decision support systems used by human controllers in the AOCC, two factors have continued to limit the performance of the disruption resolutions that are applied. First, the decision support systems do not explicitly proffer solutions to specific schedule disruptions, and as such, human controllers in the AOCC are required to be reactive in addressing disruptions by using their best judgement based upon their prior experience from resolving the same (or similar) disruption. Second, majority of the decision support

(computer) systems used by multiple departments in an airline (including the AOCC) and other air transportation stakeholders (e.g. airports) are not designed or developed at the same time nor by the same vendor [24]. As such, information and data are required to be entered into multiple computer systems thereby exposing human controllers in the AOCC to data input problems and errors. As such, information entered into a decision support system for disruption management may be out of sync with other systems and yield incorrect decisions due to lack of data integrity.

In the bid to improve data integrity for existing decision support systems, airlines have significantly invested in creating better localized data collection platforms within their respective organizations, which can amass information from different sources within and outside the organization that is easily accessible through a centralized data server [47]. As such, there is a need to fully leverage the ubiquity and accessibility of information (data) collected by existing platforms in the AOCC to enhance agile decision-making capabilities of the AOCC during airline disruption management. To this effect, this chapter provides a comprehensive discussion on exploratory analysis administered on historical scheduling and operations recovery data supplied by a major airline in the United States, which serves as the basis for the development of the predictive and prescriptive models discussed in subsequent chapters of this dissertation.

The next section in this chapter provides an overview of the elements of historical scheduling and operations recovery data retrieved from a major U.S. airline, followed by a section that expansively discuses several relevant interrelated processes for exploratory data analysis.

3.1 Data Overview

The raw data utilized for demonstrating the exploratory analysis discussed in this chapter and ultimately creating the *i-MAS* framework, which is the focus of this dissertation, was provided by Southwest Airlines. Like many major U.S. airlines,

Southwest Airlines employs an integrated AOCC organization wherein all functional roles share the same physical space (at the airline's headquarters in Dallas) and are hierarchically dependent on AOCC Supervisors for multiple problem dimensions in airline operations recovery [48]. As the largest carrier in the United States in terms of originating domestic passengers boarded with more than 4,100 flight schedule operations daily to over 100 destinations, the supervisors (and controllers) at Southwest Airlines Network Operations Control (SWA-NOC) seek to use technology to see the impact of their decisions to make better ones for improved disruption management. For many years, the controllers at SWA-NOC relied on gut instincts to track and understand how their disruption resolution actions cascaded throughout the airline's network, but could not inform their instincts with data. To address this issue, upper management at SWA-NOC created the Baker workgroup; an integrated team of supervisors and software developers dedicated to improving decision-making during disruption management by developing and enhancing a suite of computerized decision support systems called the Baker tool. In order to better support the Baker tool, the workgroup created an autonomous data collection platform to record flight schedules that are subject and not subject to different disruption incidents in the Southwest Airlines route network.

As such, the raw data generously provided to us for the research discussed in this dissertation contains approximately 1.1 million instances of direct flight schedules from the Southwest Airlines route network operations recorded from September 2016 to September 2017; of which there are 620,000 flight schedules that were not subject to disruptions, over 430,000 flight schedules that were subject to flight delays, and approximately 26,000 flight schedules that were either cancelled or diverted. The instances of disrupted flight schedules (i.e. delayed, cancelled or diverted flight schedules) are distributed across eleven separate functional roles in SWA-NOC (i.e. the AOCC) that represent primary disruption resolution domains for different problem dimensions in airline disruption management. Table 3.1 reveals a list of functional disruption resolution domains in the Southwest Airlines Network Operations Control,

Table 3.1. Disruption outlook for functional domains in Southwest Airlines network operations control from September 2016 to September 2017

Functional Domain	**Affected Problem Dimension**	**Disruption Class**	**Delayed Flight Schedule Instances**	**Cancelled Flight Schedule Instances**	**Diverted Flight Schedule Instances**
Customer Hold	Aircraft and Passenger	Controllable	46,870	0	0
Dispatch CSC	Aircraft and Crew	Controllable	17,468	0	0
Flight Operations	Crew	Controllable	36,370	1099	909
Fuel Management	Aircraft	Controllable	4,841	0	0
Ground Operations	Aircraft and Passenger	Controllable	168,375	2518	460
Inflight	Crew	Controllable	79,444	984	67
Maintenance	Aircraft	Controllable	33,518	55	0
NAS	All	Uncontrollable	22,644	2619	433
Security	Passenger	Controllable	2,955	6	11
Technology	All	Controllable	8,953	0	0
Weather	All	Uncontrollable	12,659	12,156	4905

including the corresponding problem dimensions they seek to address, and the class of disruption and the number of instances of different effects of a disruption class for a specific functional domain. The disruption class defines the origination of a specific disruption, and as such, disruptions resolved by functional domains in SWA-NOC with a "controllable" disruption class indicate that all instances of disrupted flight schedules associated with those domains were caused (or could have been avoided) by the airline. Conversely, disruptions resolved by functional domains in SWA-NOC with an "uncontrollable" disruption class indicate that all instances of disrupted flight schedules affiliated with those domains were not caused (nor could have been avoided) by the airline. A brief description of the functional disruption resolution domains (or roles) in SWA-NOC highlighted in Table 3.1 is as follows:

1. ***Customer Hold***: This functional domain addresses disruptions related to holding aircraft for passengers on inbound flight connections and holding aircraft to accommodate passengers off cancelled and delayed flights. As such, the customer hold functional domain resolves the aircraft and passenger problem dimensions in airline disruption management. Disruption instances for the customer hold domain accounted for about 11% of delayed flight schedules in the Southwest Airlines route network over the one-year period (i.e September 2016 to September 2017).

2. ***Dispatch CSC***: This functional domain manages disruptions related to flight dispatch activities by the airline that also includes holding flights to accommodate international flight schedule slot times. To that effect, the Dispatch CSC functional domain addresses the aircraft and crew problem dimensions during disruption management, and disruption instances related to Dispatch CSC represented 4% of delayed flight schedules in the airline operations between September 2016 and September 2017.

3. ***Flight Operations***: This functional domain resolves disruptions defined by Pilot (cockpit crew) scheduling activities as they relate to Pilot tardiness and

normal aircraft readiness, and addresses the crew problem dimension of airline disruption management. Between September 2016 and September 2017, disruption instances related to Flight Operations represented about 8.5% of delayed flight schedules, 5.7% of cancelled flight schedules, and 13.5% of diverted flight schedules in Southwest Airlines operations.

4. ***Fuel Management***: This functional role in SWA-NOC manages disruptions related to aircraft fueling and other energy administration activities, and addresses the aircraft problem dimension during disruption management. Disruption instances related to Fuel Management between September 2016 and September 2017 represented 1.1% of delayed flight schedules in Southwest Airlines operations.

5. ***Ground Operations***: This functional domain in SWA-NOC manages disruptions defined by several activities ranging from passenger boarding and aircraft provisioning to ramp services and aircraft towing, and as such, resolves the aircraft and passenger problem dimensions in airline disruption management. Over the one year period of airline operations, disruptions related to Ground Operations accounted for the largest percentage of total flight schedule delays of 39%, and the third highest percentage of total flight schedule cancellations of 13%.

6. ***Inflight***: Similar to Flight Operations, Inflight resolves disruptions defined by Flight Attendant (cabin crew) scheduling activities as they relate to Flight attendant tardiness and normal aircraft preparedness, and thus addresses the crew problem dimension of airline disruption management. Between September 2016 and September 2017, disruption instances related to Flight Operations represented about 18.5% of delayed flight schedules, 5.1% of cancelled flight schedules, and about 1% of diverted flight schedules in Southwest Airlines operations.

7. ***Maintenance***: This functional domain resolves disruptions defined by aircraft maintenance and inspection activities, and as such, addresses the aircraft problem dimension of airline disruption management. Disruption instances related to Maintenance represented 7.8% of delayed flight schedules and about 0.3% of cancelled flight schedules during Southwest Airlines operations from September 2016 to September 2017.

8. ***NAS***: This adopted functional role in SWA-NOC manages disruptions defined by air traffic control activities related to gate hold for congestion at departure and arrival airport stations. As such, the NAS functional domain addresses uncontrollable disruptions representing all problem dimensions during airline disruption management. Disruption instances associated with NAS represented 5.3% of delayed flight schedules, 13.5% of cancelled flight schedules and 6.4% of diverted flight schedules during Southwest Airlines operations from September 2016 to September 2017.

9. ***Security***: This functional domain addresses disruptions defined by security measures enforced to ensure the safety and convenience of passengers at airports prior to aircraft boarding. Its responsibilities includes managing disruptions due to baggage screening by TSA (Transportation Security Administration) at the skycap or ticket counter. As such, the Security functional domain resolves the passenger problem dimension during airline disruption management. Between September 2016 and September 2017, disruption instances related to Security represented the least percentages of total delayed, cancelled, and diverted flight schedules of 0.7%, 0.03%, and 0.16%, respectively, in Southwest Airlines operations.

10. ***Technology***: This functional role manages all disruption activities defined by system-wide technology outages, and thus aims to resolve all problem dimensions during airline disruption management. Disruption instances related to

Technology accounted for 2.1% of all delayed flight schedules in Southwest Airlines operations between September 2016 and September 2017.

11. ***Weather***: Similar to NAS, this adopted functional domain in SWA-NOC manages all kinds of uncontrollable disruption defined by inclement weather activities. To that effect, the Weather functional role aims to resolve the aircraft, crew and passenger problem dimensions during disruption management. Disruption instances associated with the Weather functional domain accounted for the highest percentage of cancelled and diverted flight schedules (62.6% and 72.8% respectively) among all functional domains in SWA-NOC between September 2016 and September 2017. In addition, delayed flight instances related to Weather represented 2.9% of the total delayed flight instances addressed by all functional domains in SWA-NOC over the one year data collation period.

3.2 Data Analysis

The previous section provided a macroscopic overview of disruption activities for different functional roles in SWA-NOC and revealed that about 42% of all flight schedules for Southwest Airlines route network operations from September 2016 to September 2017 were disrupted. As a result, the functional disruption resolution domains in SWA-NOC were most likely to address irregular operations due to delayed flight schedules, which represent approximately 94% of all disrupted flight schedules recorded from September 2016 to September 2017. Furthermore, there are two separate chunks of data which are defined by the occurrence of disruption during flight schedule execution. The first chunk, which is known as the *non-disrupted* data set, represents the larger chunk that contains instances of flight schedules that executed without any disruption. The smaller chunk, also known as the *disrupted* data set, contains instances of flight schedules that executed with disruption, and thus represent instances of flight schedule execution due to irregular operations. As such, the major difference between the *non-disrupted* data set and *disrupted* data set is the existence

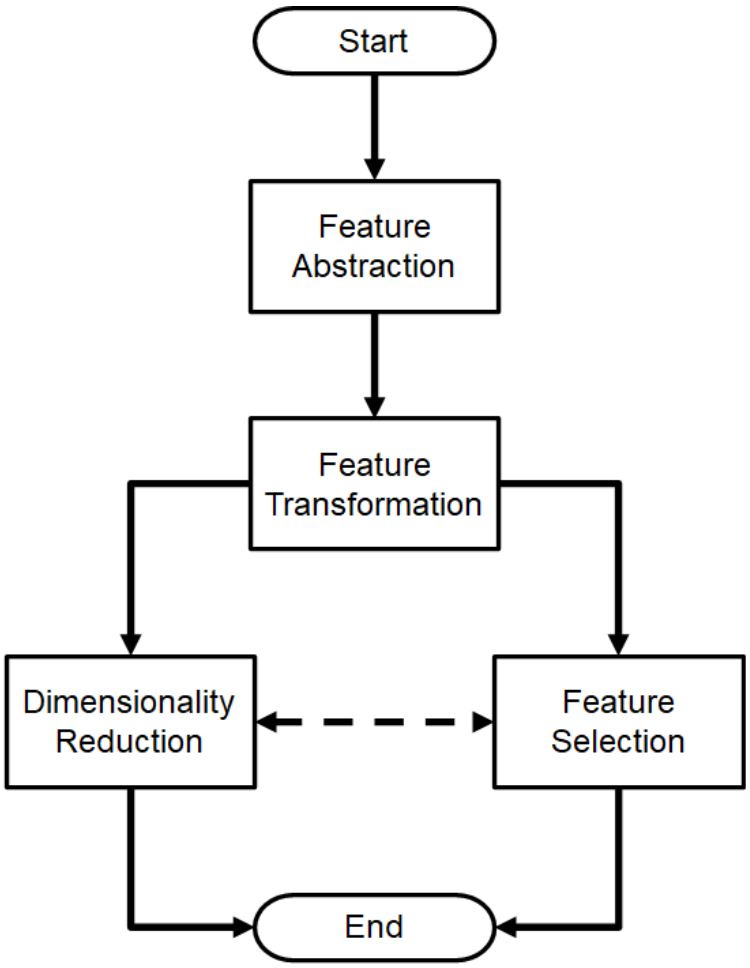

Figure 3.1. Data analysis procedure for a *SIR* design paradigm

of additional data features (i.e. disruption features) in the *disrupted* data set that indicate different types of disruption. However, there is a small subset (6%) of the *disrupted* data set that represents instances of canceled and diverted flight schedules, which have less data fields with sparse data entries that present significant challenges for machine learning applications. To that end, we restrict the scope of the research presented in this dissertation to irregular operations based upon delayed flight schedules and ignore flight cancellations and diversions which are primarily limited to the Weather functional domain. Henceforth, irregular operations only represent controllable and uncontrollable disruptions due to delayed flight schedules.

Fine details that highlight pertinent high-level patterns for elements (or features) that define each flight schedule in the raw data set introduced in Section 3.1 can not be readily observed nor acknowledged through macroscopic inspection. Hence, in order to mine relevant microscopic information from raw data, this section elucidates the exploratory data analysis used to effectively generalize pattern-finding schemes for consistent flight schedule features that are applicable in all functional roles in SWA-NOC. Fig 3.1 shows the general procedure adopted for performing exploratory data analysis on the raw data set. The process commences by abstracting separate data features that represent distinct properties of flight scheduling and operations, next raw data features are transformed into data forms that are readily decipherable by appropriate machine learning algorithms. The data transformation is necessary for applying separate methods for identifying critical data features and reducing the dimension space of the data set achieved by the feature selection and dimensionality reduction processes shown in Fig 3.1. The subsequent parts of this section provide more insight into the aforementioned processes as they relate to the historical scheduling and operations data adapted for the research discussed in this dissertation.

3.2.1 Feature Abstraction

Feature abstraction (often referred to as data abstraction) is an effective technique for accommodating semantic relationships between features in a database [49]. Feature abstraction ensures that features that define specific properties of fundamental tenets of flight scheduling and operations (embedded in each flight schedule from the raw data set) is generalized into abstract values [35]. As such, a specific property can be viewed as a specialized quality of an abstract value. The properties describing the tenets of flight scheduling and operations are described by two separate principles of abstraction that represent a knowledge abstraction [35, 50, 51]. These related feature abstraction principles exist for creating semantic associations among data features

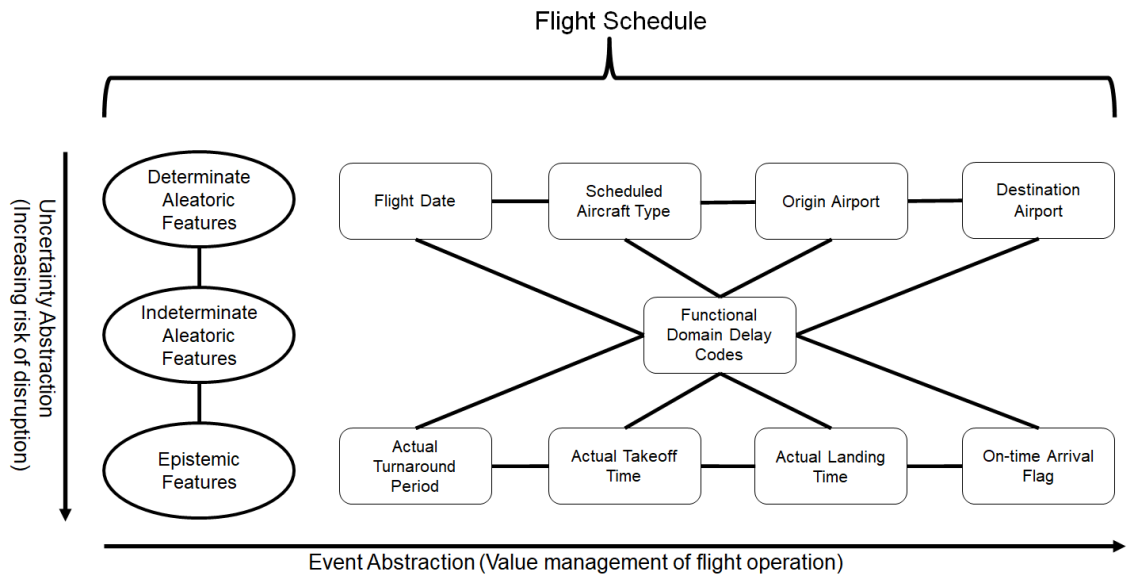

Figure 3.2. Sample knowledge abstraction of a basic flight schedule for airline disruption management

for airline scheduling and disruption management and are namely: *event abstraction* and *uncertainty abstraction*.

- *Event abstraction*: Event abstraction serves two primary purposes. First, it characterizes the importance of planned airline activities and resources associated with a specific flight schedule, and how their importance can be subject to change prior to (or on) day of operation due to the risk of disruption from passenger-boarding at a departure station to aircraft gate-parking at an arrival station. Second, event abstraction defines the manner in which planned airline activities for a specific flight schedule varies during schedule execution based upon the impact of irregular operations. As such, the event abstraction principle is synonymous to *flight operation value abstraction*, wherein the specific value (or profit) that a particular flight schedule in the airline route network provides is appraised by how effectively the flight schedule features estimated prior to schedule execution align with flight schedule features that are realized during and after schedule execution. To that effect, flight schedule features

that are determined prior to schedule execution represent low-value features for event abstraction, and flight schedule features estimated during and after schedule execution represent high-value features for event abstraction.

- *Uncertainty abstraction*: From an airline planning perspective, the uncertainty abstraction principle, also analogous to a *functional domain planning abstraction*, defines the manner in which the uncertainty for the risk of irregular operations during schedule planning and execution is quantified and propagated via various features in the data set. Most airlines, including Southwest Airlines, adopt a perspective on the scheduling process that accentuates the internal planning approach of different planning departments as an iterative cycle of flight schedule development and assessment over a timeline horizon, such that the flight schedule is continually adjusted and optimized until a suitable schedule is obtained or the planning period is over [3]. Thus, the primary purpose of *uncertainty abstraction* is to express the relationships among flight schedule features that are representative of the transitions through three iterative and interconnected airline planning phases namely: strategic planning, tactical planning and operational planning respectively [6]. Strategic planning, otherwise known as "future scheduling", focuses on long-term decision-making for the subsequent tactical and operational planning phases, such that a generic service plan consisting of essential and viable sets of serviceable routes without specific aircraft and crew assignments and tentative departure and arrival times are determined. Tactical planning or "current scheduling" focuses on creating a refined schedule of operations for the service plan based upon the resources that are actually expected to be available to the airline over a definitive time period. Lastly, the operational planning phase focuses on adjusting the schedule generated in the tactical planning phase with respect to changes in demand for air travel prior to executing the flight schedule, and also on executing the schedule with minimal penalty (cost) in the event of unexpected disruptions on the day of operation (i.e. rescheduling for disruption management) [52].

By applying the event and uncertainty abstraction principles, we identify three separate classes of features in the raw data that can be used to enable robust airline disruption management and are described as follows:

1. *Determinate aleatoric features*: These represent flight schedule features that are determined during the strategic planning phase of airline planning and are required to remain unchanged during schedule execution on the day of operation. Examples of determinate aleatoric features include flight date, origin (departure) station, destination (arrival) station, route originator indicator, route distance, etc. With respect to disruption management, determinate aleatoric features represent flight schedule features whose alternatives do not differ considerably each time the AOCC invokes a disruption management initiative. Thus, from a statistical perspective, determinate aleatoric features are flight schedule features that are subject to the least possible uncertainty for the risk of reassessment (or alteration) during irregular operations for disruption management, based upon inherent randomness of disruption events. For instance, airport identifiers and exact longitude and latitude coordinates that provide specific information for origin and destination stations are always assumed to remain unchanged, by the AOCC, during the recovery of a delayed flight schedule. However, if a human specialist in the AOCC chooses to divert the same delayed flight to another airport during schedule execution, then the airport information for the destination station changes to that of the airport where the flight is to be diverted. It is important to note that this scenario is unlikely, based upon our research scope, because we consider irregular operations for delayed flight schedules only.

2. *Indeterminate aleatoric features*: These are separate data features from flight schedule features that represent disruption types for different functional domains, which occur randomly during schedule execution on day of operation. Examples of indeterminate aleatoric features include delay codes for uncon-

trollable inclement weather and controllable maintenance inspections. From a disruption management perspective, indeterminate aleatoric features represent triggers for the need of the AOCC to address a specific disruption. As such, indeterminate aleatoric features are data features that can create the most uncertain responses in disruption management initiatives employed by the AOCC during schedule execution. From a statistical perspective, indeterminate aleatoric features are data features that are subject to the most possible uncertainty for the risk of occurrence (or instantiation) of irregular operations during schedule execution, due to inherent randomness of disruption events. For example, inclement weather at a particular airport may require a human specialist in the AOCC to delay the departure of a specific flight at the (origin) airport and reassign some or all of its passengers to another flight with a later departure, while also reallocating the arrival of the original delayed flight to a different gate at the destination airport.

3. *Epistemic features*: These represent flight schedule features that are determined during the tactical and operational phases of airline planning and can be subject to change during schedule execution on day of operation. Examples of epistemic features include specific departure and arrival times during the day, aircraft type, delay periods, actual turnaround and block time periods. With regards to disruption management, epistemic features represent flight schedule features with considerable amount of alternatives for every time the AOCC initiates a disruption management plan. As such, from a statistical standpoint, epistemic features are flight schedule features that are subject to the most possible uncertainty for the risk of alteration during irregular operations for disruption management, due to lack of knowledge of the exact impact of their alteration. For instance, following a specific disruption like late arrival of flight crew for a scheduled flight, a human specialist in the AOCC may choose to delay the departure of the flight by a specific period of time after the original departure time. However, most times, the human specialist can not guarantee that the

decision on applying a particular delay duration after scheduled departure will produce a specific recovery plan, due to the cascading effect of disruptions in large airline networks.

Fig. 3.2 shows a generic knowledge abstraction for airline disruption management based upon some specific flight schedule features. The horizontal axis in Fig. 3.2 represents event abstraction for defining the value of flight operations management based upon the perishable nature of a flight service during schedule execution, while the vertical axis represents uncertainty abstraction for defining the risk of disruption instances and schedule alteration during flight schedule planning.

3.2.2 Feature Transformation

While the abstraction of raw flight schedule data features provides an excellent avenue for effectively representing latent planning capabilities in airline operations control, the quality of the knowledge extracted from the raw data can be enhanced through transformation to enable discernible representation and interpretation for machine learning algorithms [53, 54]. These algorithms provide efficient means for easily recognizing useful patterns and relationships amongst flight schedule features in a data set. To this end, feature transformation is the process of converting flight schedule and disruption features in raw historical airline scheduling and operations data into relevant mathematical properties (or functions) that can be readily understood by machine learning algorithms. Every direct flight schedule in the raw data set is defined by forty separate data features (or attributes) that describe different resources, behaviors, and performance indicators that are observable during airline scheduling and disruption management. As such, raw flight schedule features can be separated into four distinct categories namely: geographical features, temporal features, categorical features, and continuous features.

- *Geographical features*: These are flight schedule features which represent resources, behaviors, or performance indicators that require or enable the percep-

tion and property of geographic location (or position) during airline disruption management. Examples of geographical features in the raw data set are International Air Transport Association (IATA) codes for departure and arrival airport stations, and the identifier for the origin of the first departure flight of the day.

- *Temporal features*: These are flight schedule features that describe and enable the perception and property of time during airline scheduling and disruption management. Temporal features are conceptualized by four different types of time [55] namely:

  1. *ordinal time*: This represents time points that occur one after another on day of operation. Examples of flight schedule features defined by ordinal time are time-of-day events such as aircraft pushback time, takeoff time, landing time, and aircraft gate-parking time.

  2. *interval time*: This represents time events that are measured on an interval scale with a specific duration (or length). Examples of flight schedule features characterized by interval time include the duration between aircraft pushback and aircraft gate-parking otherwise known as blocktime, the duration for boarding passengers and loading cargo unto an aircraft also known as turnaround, and the duration of any form of delay in airline operations during schedule execution.

  3. *cyclic time*: This describes cyclic or repeatable processes wherein the application of an ordered relation is inane. Flight date is an example of a flight schedule feature characterized by cyclic time.

  4. *branching time*: This represents time points that can occur in different branches or alternatives to describe several scenarios or processes. Thus, all temporal flight schedule features in the raw data set are defined by branching time.

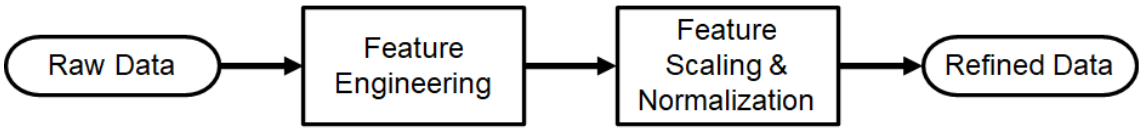

Figure 3.3. Feature transformation process of raw data for airline disruption management

- *Categorical features*: These are flight schedule features that represent fields in the raw data defined by discrete values which belong to a finite set of categories or classes. Categorical features can be text or numeric, and are separated into two classes namely nominal and ordinal, based upon the perception of ordering.

  1. *nominal*: Nominal categorical features represent flight schedule features for which there is no concept of ordering among different values of each feature. An example of a nominal categorical feature is A0, which is a binary number (i.e. 0 or 1) indicating whether or not a flight schedule arrives exactly on time.

  2. *ordinal*: Ordinal categorical features represent flight schedule features for which there is a strict adherence to the concept of ordering among different values of each feature. An example of an ordinal categorical feature in flight schedule data is aircraft type, which effectively characterizes the relevance of size and seat capacity for aircraft performance.

- *Continuous features*: These are flight schedule features that represent fields in the raw data, which have infinitely many alternatives between any two values. Examples of continuous features in raw flight schedule data include digital timestamps for different time-of-day events (such as takeoff time) during schedule execution.

Fig 3.3 reveals a two-layer data transformation process of raw flight schedule data features for airline disruption management. The first layer, known as feature

engineering, enables the creation of additional data features from mathematical functions that characterize rudimentary properties of the airline operations control center. Next, these data features are combined with extant low-level data features in the raw data set and normalized by using fundamental statistical parameters in the second layer through a process called feature scaling.

(i) **Feature engineering** represents a *first-degree* transformation of raw flight schedule data that defines the augmentation of properties associated with daily routines of functional roles in the AOCC by using mathematical principles. Thus, flight schedule features representing geographic locations (i.e. geographical features) such as departure and arrival stations are first transformed into spherical directional vectors based upon the longitude and latitude coordinates of their corresponding airport stations, and subsequently transformed into the distance between the departure and arrival airports on an oblate spheroid Earth via the Vincenty geodesic equation [56]. Ordinal temporal flight schedule features (such as departure and arrival times) are transformed into two separate periodic (i.e. sine and/or cosine) vectors of different amplitudes, based upon a 24-hr clock period and the percentage of 8-hr work shift completed (at the time of departure or arrival) by human specialists in the AOCC, respectively. The work shift characterization of time-of-day events, via a periodic vector, is intended to capture and represent daily disruption resolution proclivities of human specialists, which can be induced by how much time the specialists have to address a disruption before their work shift is complete. Cyclic temporal flight schedule features defined by Gregorian dates are transformed into four separate periodic vectors, whose periods are based upon the season of the year, month of the year, day of the week, and day of the year respectively.

Lastly, most categorical flight schedule features defined by texts are transformed into binary numbers by one-hot encoding [57], wherein all n feature values are represented as a n-dimensional sparse vector with zero entries except for one of the dimensions for which the entry is one. However, categorical flight

schedule feature values for aircraft type, which are defined by aircraft model codes, are transformed into discrete numbers based upon the total number of available seats in the aircraft. A secondary objective of feature engineering is to provide a precursor for continuous feature representation by ensuring that all data features are in numeric form. As such, feature engineering may not be applicable to flight schedule features that are already in continuous form in a raw data set.

(ii) **Feature Scaling** represents a *second-degree* transformation of raw numeric data features that include additional features created from feature engineering, such that a uniform statistical grounding basis is used to transform values for all data fields (i.e. flight schedule features) in the raw data set into bounded continuous values that describe a differentiable function. We explore three different approaches for enabling feature normalization [58] namely: *Standard scaling*, *Range scaling*, and *Power scaling*.

(a) *Standard scaling*: Standard scaling or standardization normalizes values for each flight schedule feature in the data set by removing the mean of the values and scaling to a unit variance, thus resulting in a standard score for each value. The standard score, z_i, of an arbitrary sample (i.e data feature value), x_i, in the data set is calculated as follows:

$$z_i = \frac{(x_i - u)}{s} \tag{3.1}$$

where u and s represent the mean and standard deviation, respectively, of all values for each flight schedule feature in the data set. As such, standardization provides a platform to ensure that each flight schedule feature in the data set follows a Gaussian distribution with zero mean and a variance of one.

(b) *Range scaling*: Range scaling, otherwise known as min-max normalization, transforms each flight schedule feature in the data set by scaling the values

of each feature by the difference between the maximum value and the minimum value (i.e. range). This results in adjusted values of a range (or distance) between zero and one for each flight schedule feature. The adjusted (range-scaled) value, y_i, of a characteristic flight schedule feature in the data set is calculated as follows:

$$y_i = \frac{x_i - \min(X)}{\max(X) - \min(X)} \tag{3.2}$$

where x_i and X represent an original value and set of all original values, respectively, for a flight schedule feature.

(c) *Power scaling*: Power scaling involves adapting a family of parametric and monotonic transformations to convert flight schedule data values from any distribution to the closest possible representation of Gaussian distribution, so as to reduce variance and skewness in data. An appropriate power transformation of flight schedule and disruption features is the Yeo-Johnson transform [59], because it can be applied to all forms of numeric data just like standard and range scaling transforms. The Yeo-Johnson transform is given by:

$$x_i^{(\lambda)} = \begin{cases} [(x_i+1)^{\lambda} - 1]/\lambda & \text{if} \quad \lambda \neq 0, x_i \geq 0, \\ \ln(x_i+1) & \text{if} \quad \lambda = 0, x_i \geq 0, \\ -[(-x_i+1)^{2-\lambda} - 1]/(2-\lambda) & \text{if} \quad \lambda \neq 2, x_i < 0, \\ -\ln(-x_i+1) & \text{if} \quad \lambda = 2, x_i < 0 \end{cases} \tag{3.3}$$

where x_i and λ represent an original data value and an arbitrary parameter that is determined through maximum likelihood estimation [60], respectively.

Completion of the feature transformation process shown in Fig. 3.3 results in a refined, continuous data set that can be readily comprehensible by suitable machine learning estimators.

3.2.3 Dimensionality Reduction

The efficacy of constructing and applying relevant machine learning algorithms for identifying and acknowledging high-level properties from data features (such as flight schedule and disruption data features) is dependent on the form in which the data values are presented. To this end, feature transformation has a strong propensity to increase the number of elements in the flight schedule and disruption feature space that constitutes the problem dimensions for airline disruption management. As such, the intrinsic dimensionality of the refined data appropriated for airline disruption management is defined by the least number of flight schedule and disruption features required to delineate observed behavioral properties from AOCC routines. Hence, dimensionality reduction is the process of mitigating the curse of dimensionality [61] and other unwanted properties of high-dimensional feature space through classification, visualization, and compression of high dimensional data obtained as a result of feature transformation [62]. In essence, dimensionality reduction aims to provide a rudimentary means to attain and observe the latent feature space of a refined data set for airline disruption management.

From a mathematical perspective, we assume that the refined flight schedule and disruption data set is represented in a $n \times m$ matrix $\mathbf{X}$, which consists of n feature vectors $\mathbf{x}_i (i \in \{1, 2, ..., n\})$ with dimensionality m. Furthermore, we assume that the refined data set has an intrinsic dimensionality d, such that $d < m$ and often $d << m$. The intrinsic dimensionality property refers to points in the refined data set $\mathbf{X}$, which lie near a manifold with dimensionality d that is embedded in the m-dimensional feature space. To that effect, dimensionality reduction techniques transmute the refined flight schedule and disruption data set $\mathbf{X}$ with dimensionality m into a new data set $\mathbf{Y}$ with dimensionality d, while maintaining the geometry of the refined data set $\mathbf{X}$ as much as possible. Typically, the intrinsic dimensionality d and the geometry of the manifold of the new data set $\mathbf{Y}$ are unknown, and as such, most dimensionality reduction techniques require that certain assumptions about the

properties (like intrinsic dimensionality) of the refined data set be made a priori. For the remainder of this section, we denote a high dimensional data instance for flight schedule and disruption (i.e. datapoint) by $\mathbf{x}_i$, such that $\mathbf{x}_i$ is the ith row of the refined m-dimensional data set $\mathbf{X}$. In complement, the low-dimensional equivalent of $\mathbf{x}_i$ is expressed by $\mathbf{y}_i$, where $\mathbf{y}_i$ is the ith row of the new d-dimensional matrix $\mathbf{Y}$.

To demonstrate the usefulness of dimensionality reduction on refined flight schedule and operations data, we investigate two separate techniques that employ linear and nonlinear principles nicknamed *PCA* and *t-SNE*, respectively, by utilizing delayed flight schedule and disruption instances for the Weather functional domain in SWA-NOC between September 2016 and September 2017. It is important to note that the flight schedule data for the Weather functional domain constitutes a subset (with 12,659 delayed flight schedule instances) of the full refined data set. For validation, we adopt min-max normalization for scaling all feature values in the refined data set because both dimensionality reduction techniques strongly depend on Euclidean distances between refined high-dimensional datapoints $\mathbf{x}_i$ and $\mathbf{x}_j$ to obtain and simplify the gradient of their respective cost functions [62, 63].

1. *Principal Component Analysis (PCA)*: Principal component analysis or *PCA* is a standard non-parametric tool in modern data analysis used for extracting relevant information from large and confusing data sets [64]. *PCA* is also a full spectral linear technique for dimensionality reduction that embeds data into a linear subspace of lower dimensionality. In the lower dimension, the refined variables (or data features) in the data set are transformed into linear combinations of the data features, which are called principal components. With minimal effort, *PCA* provides a schema for reducing a fairly complex data set to a lower dimension in order to show simplified structures that often define it, by revealing as much of the variance in the data as possible. As such, the first and second principal components are the orthogonal linear combinations of the refined data features that have the largest and second-largest possible variance (or inertia), respectively, in the refined data set. In mathematical

terms, *PCA* aims to find a linear mapping $\mathbf{X}$ that maximizes the cost function defined by $\text{trace}(\mathbf{S}^T \text{cov}(\mathbf{X})\mathbf{S}))$, wherein $\text{cov}(X)$ is the sample covariance matrix of the refined data [65]. Thus, the linear mapping created by the d principal components (or principal eigenvectors) are solutions to the eigenproblem defined as follows:

$$\text{cov}(\mathbf{X}) = \lambda(\mathbf{S}) \tag{3.4}$$

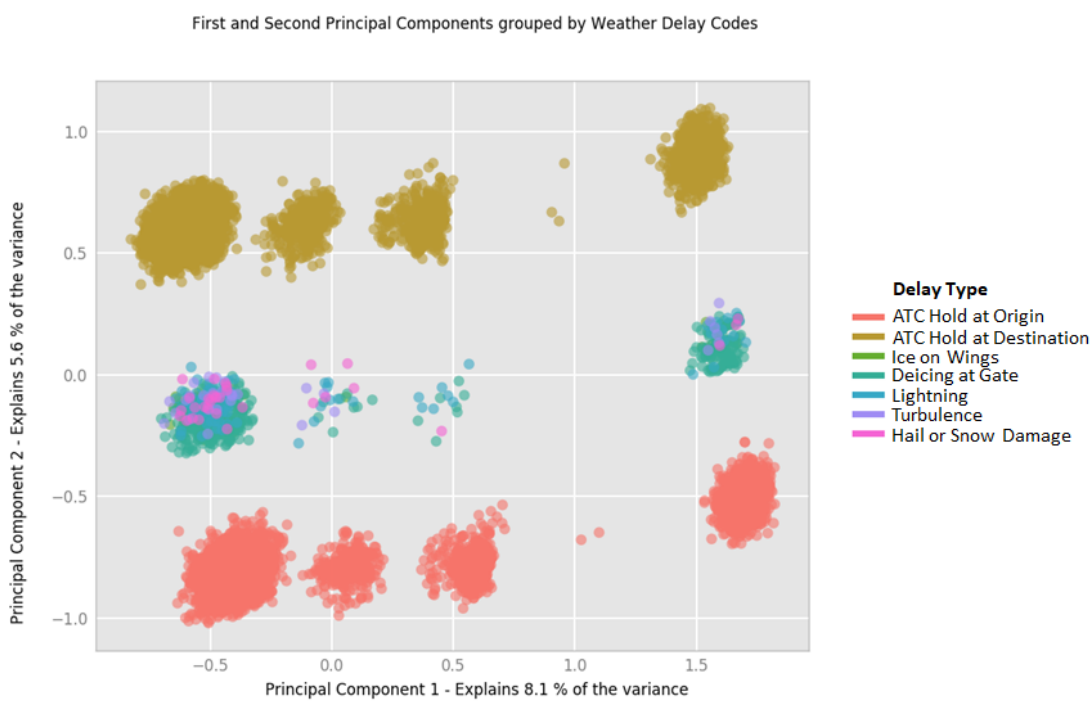

Figure 3.4. Principal component analysis of indeterminate aleatoric features for Weather domain

The lower dimensional representation of the refined flight schedule and disruption feature instances, defined by $\mathbf{y}_i$ of $\mathbf{x}_i$ datapoints, are computed by mapping them onto a linear basis $\mathbf{Y} = \mathbf{XS}$ that solves the eigenproblem for the d principal eigenvalues defined by λ, via the scikit-learn software [58].

Fig. 3.4 presents a visualization of the analysis for the first two principal components describing indeterminate aleatoric features that represent delays for the Weather functional domain in SWA-NOC. The first and second principal components represent orthogonal linear combinations of the refined flight schedule features that account for 8.1% and 5.6%, respectively, of the variance of indeterminate aleatoric features related to weather delays in the data set. Fig. 3.4 reveals that there are four major types of weather-related delays (*ATC Hold at Origin*, *ATC Hold at Destination*, *Deicing at Gate*, and *Hail or Snow Damage*) in the data set. *ATC Hold at Origin* and *ATC Hold at Destination* represent weather delays due to gate hold from air traffic control (ATC) at departure and arrival stations respectively. *Deicing at Gate* and *Hail or Snow Damage* represent weather delays due to deicing at the gate, and aircraft swap due to hail or snow damage respectively. Fig. 3.4 shows that the data set is divided into four separate clusters of the same delay type along the axis of the first principal component. This reveals that the axis of the first principal component (horizontal) represents linear combinations of flight schedule features that capture the seasonal behavior of weather-related delays, as each data cluster describes each weather season over the one-year data-collation period. Furthermore, the data set is divided into two polarizing (*ATC Hold at Origin* and *ATC Hold at Destination*) and two overlapping (*Deicing at Gate* and *Hail and Snow Damage*) clusters along the second principal component (vertical) axis. This shows that the axis of the second principal component represents linear combinations of flight schedule features that capture the difference in the types of indeterminate aleatoric features for weather-related delays in the refined data set.

The patterns and information gleaned from the results and observations from the principal component analysis (*PCA*) method can be appropriated for informing model development for airline disruption management. For instance, the seasonal relationship among the four predominant types of weather-related delays observed in Fig. 3.4 can be quantified via a linear combination of flight

schedule features, which suggests that decision-making by human specialists in the AOCC is sensitive to weather seasons. In addition, the overlapping effect observed between *Deicing at Gate* and *Hail and Snow Damage* in Fig. 3.4 is representative of similarities in the type of disruption resolutions used for weather-related delays during the winter season, thus bolstering the significance of the effect of seasonal properties on decision-making by human specialists in the AOCC.

2. *t-distributed Stochastic Neighborhood Embedding (t-SNE)*: t-distributed Stochastic neighborhood embedding or *t-SNE* represents a recent advancement in clutering and visualization for dimensionality reduction that provides a nonlinear platform for transforming the Euclidean distances between refined values (i.e. datapoints) of flight schedule and disruption features into conditional probabilities that define similarities. As such, the similarity of a datapoint x_j to another datapoint x_i is the conditional probability ($p_{j|i}$) that x_i will select x_j as its neighbor if neighbors are selected in proportion to their probability density under a Student-t distribution with one degree of freedom (i.e. Cauchy distribution) centered about x_i [62, 66]. Thus, $p_{j|i}$ remains comparatively high for datapoints in close proximity and insignificant for datapoints that are substantially separated. Mathematically, the objective of *t-SNE* is to minimize the Kullback-Leibler divergence [67, 68] between a joint probability distribution defined by P in the high-dimensional feature space and a joint probability distribution defined by Q in the low-dimensional feature space. Hence, the Kullback-Leibler divergence represents the cost function of the following optimization problem, which is solved via the scikit-learn software [58]:

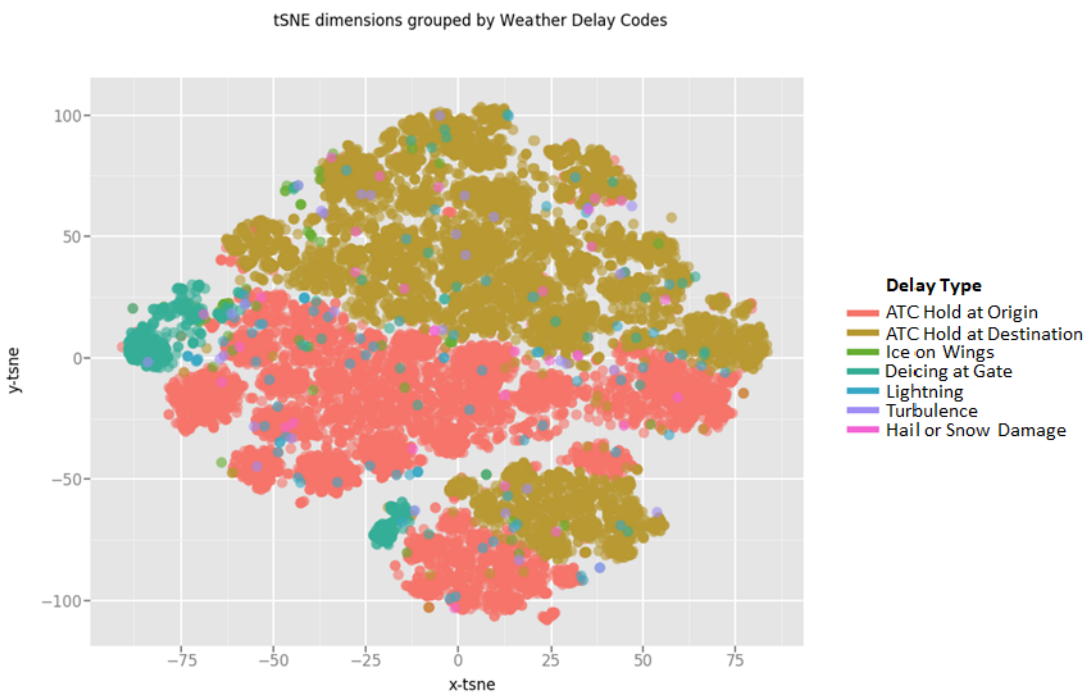

Figure 3.5. t-Distributed Stochastic Neighborhood Embedding analysis of indeterminate aleatoric features for Weather domain

$$
\begin{aligned}
\min \quad & \mathcal{KL}(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \\
\text{s.t.} \quad & p_{ij} = \frac{\exp\left(-||x_i - x_j||^2/2\sigma^2\right)}{\sum_{k \neq l} \exp\left(-||x_i - x_j||^2/2\sigma^2\right)} \\
& q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l} (1 + ||y_k - y_l||^2)^{-1}}
\end{aligned}
\tag{3.5}
$$

where p_{ii} and q_{ii} are set to zero, and $p_{ij} = p_{ji}$ and $q_{ij} = q_{ji}$ for all i, j.

Fig. 3.5 presents a strictly visual perception of the *t-SNE* nonlinear projection of the two-dimensional space for flight schedule features, which describes indeterminate aleatoric features that represent delays for the Weather functional domain in SWA-NOC. Similar to observations from *PCA*, the red and gold clusters in Fig. 3.5 reveal that weather-related delays due to *ATC Hold at Origin* and *ATC Hold at Destination* are the most prominent and oppositely related,

based upon the symmetry observed from the small and large blobs of red and gold clusters. As such, the polarizing effect observed between *ATC Hold at Origin* and *ATC Hold at Destination* in Figs. 3.4 and 3.5 can be attributed to the importance of the geographical location (i.e departure or arrival stations) on how weather-related disruption resolutions are applied in the AOCC. As such, flight schedule features associated with geographical location are relevant for creating robust models for airline disruption management.

3.2.4 Feature Selection

Although dimensionality reduction techniques provide an effective means to readily (i.e. visually) discern high-level patterns and properties associated with a data set, they are ineffectual in revealing detailed information on the specific importance of flight schedule and disruption features in a data set and their corresponding relationships [69]. To this end, feature selection presents simple fundamental methods for efficiently selecting and investigating pertinent associations among data features in a refined data set, which can provide insightful knowledge (or a priori information) for developing useful data-driven models for robust airline disruption management. In essence, feature selection methods aim to proactively enhance model prediction performances by increasing generalization (i.e. minimize data overfitting) and decreasing model runtimes [70]. There are three major categories of feature selection methods namely: wrapper, filter, and embedded methods.

Wrapper methods involve algorithms that search the feature space for plausible subsets of features by assessing each subset after running a specific model. Typically, the model is validated on a test data set to estimate the model's error rate, before a score is registered for each feature subset and the feature subset with the best score is ultimately selected. Unlike computationally intensive wrapper methods, filter methods do not consider a model when searching the feature space for relevant subsets of the feature space, and rely on general statistical measures such as Pearson correla-

tion coefficient [71] and mutual information [72]. In this manner, filter methods are somewhat analogous to dimensionality reduction techniques, such that they are not customized to a particular type of predictive model and consume significantly less computational resources than wrapper methods. Embedded methods involve feature selection methods that are entrenched in a specific learning algorithm that performs classification (or regression) and feature selection concurrently. As such, embedded methods deliver the advantages of both wrapper and filter methods with medium computational expense.

To demonstrate the relevance of feature selection on refined flight scheduling and operations data, we apply two specific types of feature selection that belong to the filter and embedded categories, respectively, to identify flight schedule features that are pertinent for disruption management during turnaround. Turnaround is an airline process (or time period) primarily representative of loading, unloading and occasional servicing of aircraft, and is crucial for minimizing overall flight schedule delays. In addition to reducing overall flight delay, most airlines typically aim to expedite the turnaround process as much as possible in order to avoid causing discomfort to passengers, stemming from long waits in the aircraft on the ground, thus invariably minimizing loss of passenger goodwill.

In that regard, the filter method that we apply is defined by mutual information and the embedded method is defined by a Gaussian process, such that actual turnaround duration is set as the target flight schedule feature. We do not consider wrapper methods in our discussion because of the significant computational expense required as compared to filter and embedded methods. Similar to the dimensionality reduction analysis discussed in Section 3.2.3, we utilize delayed flight schedule and disruption instances for the Weather functional domain in SWA-NOC between September 2016 and September 2017 for our analysis. For validation, we adopt standardization for the *second-degree* transformation (i.e. scaling) of all feature (i.e label and target) values in the refined data set, because the algorithms for both filter and

embedded methods perform best with a zero-mean Gaussian distribution as prior instantiation for each feature space in the refined data set [30, 72, 73].

1. *Mutual Information Regression (MIR)*: Mutual information is a non-negative measure from information theory that provides an excellent statistic for quantifying the degree of relatedness among flight schedule and disruption features in a refined data set. In that regard, mutual information is closely related to the entropy of a flight schedule feature based upon observing another flight schedule feature in a refined data set [74]. In addition to the ability to readily identify relationships amongst data features, mutual information provides a fundamental metric for straightforward interpretation of the relationships among data features as shared information (i.e. shannons or bits) between data features. To that effect, mutual information is insensitive to the number of instances in a data set [73].

   Mathematically, mutual information, I, is expressed as:

$$I(X;Y) = \int_X \int_Y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy \tag{3.6}$$

   where $p(x,y)$ is the joint probability density function of X and Y, and $p(x)$ and $p(y)$ are the marginal probability density functions of X and Y respectively. Thus, for feature selection, the objective is to maximize the mutual information between a subset of flight schedule features defined by $\mathbf{X}_s$ and a target flight schedule feature defined by y as represented by the following optimization problem:

$$s^* = \arg\max_s I(\mathbf{X}_s; y) \quad \text{s.t.} \quad |s| = k \tag{3.7}$$

   where k is the number of features that are to be selected. A non-parametric regression algorithm based upon entropy estimation from k-nearest neighbor distances is used to solve the NP-hard optimization problem via the scikit-learn

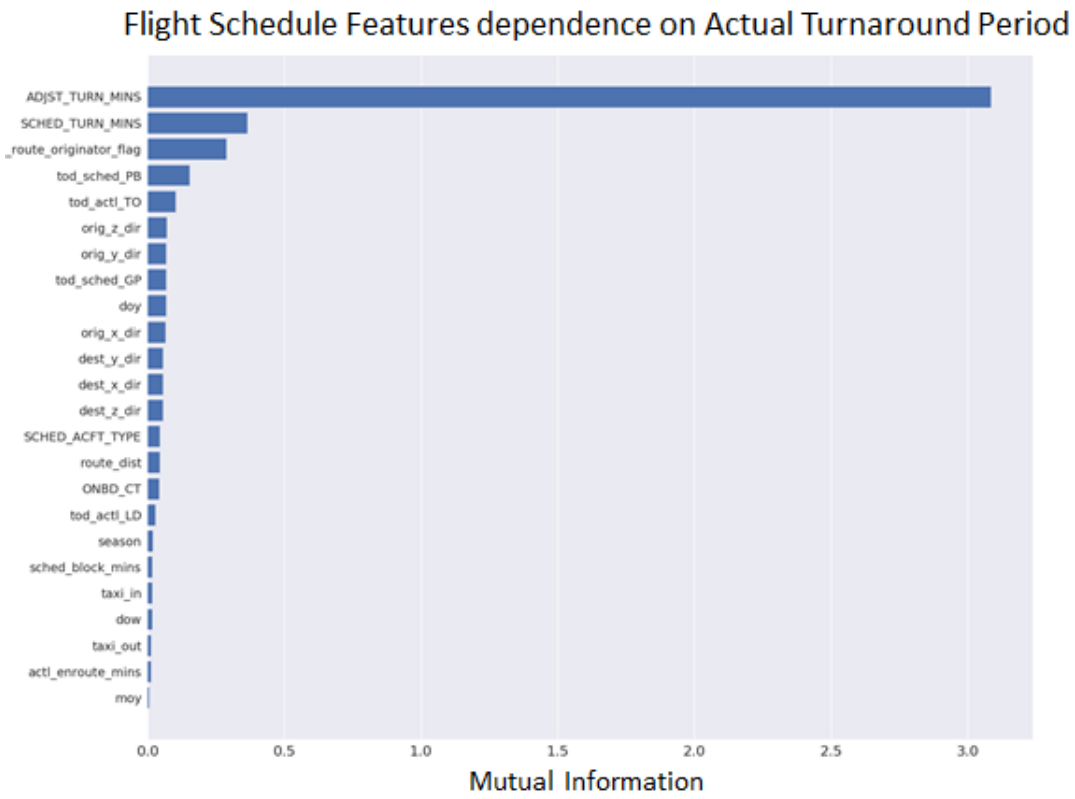

Figure 3.6. Mutual information of flight schedule data features for Weather Domain with respect to actual turnaround duration

software [58], for a set of possible combinations of data features that increases exponentially [74, 75].

Fig. 3.6 shows the mutual dependence, in decreasing order, of flight schedule features (i.e. determinate aleatoric and epistemic features) on actual turnaround period for instances of flight delay from the Weather functional domain. Fig. 3.6 reveals that turnaround duration adjusted during schedule execution has the highest mutual information of over 3, thus implying that it has the strongest mutual dependency on the decison-making for estimating actual turnaround duration during disruption management. In addition, turnaround period estimated prior to schedule execution and a flight's capacity to be a route originator

(i.e. first departure flight of the day) have the second and third highest mutual information of 0.4 and 0.3 respectively, thereby revealing a weak mutual dependency on the estimation of actual turnaround schedule for managing weather-related delay of flight schedule. Month of the year (moy) can not be selected as a significant predictor for estimating actual turnaround duration because of zero mutual dependency as shown in Fig. 3.6.

2. *Gaussian Process Regression (GPR)*: A Gaussian process is a stochastic process (i.e. random variables indexed by time or space) where a finite collection of random variables have a multivariate normal distribution [30]. As such, Gaussian Process Regression or *GPR* is the inference of continuous feature values with a Gaussian process (or distribution) prior, such that the marginal likelihood of the data is maximized [30]. Converse to *MIR*, which is a model-free method for dimensionality reduction and feature selection, *GPR* is an embedded method that offers nonlinear and non-parametric regression properties that enable the natural decomposition of flight schedule features in an airline data set for simultaneously attaining high fidelity dimensionality reduction and feature selection. *GPR* provides an appropriate medium to obtain the sensitivity and importance of flight schedule and disruption features necessary for informing the development of appropriate models for airline disruption management.

$$\begin{aligned} f(x) &\sim \mathcal{N}(0, K(\theta, x, x')) \\ \log p(f(x)|\theta, x) &= -\frac{1}{2} f(x)^T K(\theta, x, x')^{-1} f(x) - \\ &\frac{1}{2} \log \det(K(\theta, x, x')) - \frac{|x|}{2} \log 2\pi \\ &\max \log p(f(x)|\theta, x) \end{aligned} \tag{3.8}$$

GPR infers the maximum log marginal likelihood of the distribution of a target flight schedule feature for a training data set, where hyperparameters (or length-scales) associated with all flight schedule features that define different drivers of disruption management by the AOCC are optimized with respect to a certain

kernel (covariance) function. The inference problem solved by *GPR* is defined by the expressions in Eqn. 3.8. Observed flight schedule features from direct flight schedules (datapoints) in the data set are symbolized by x, while $f(x)$ symbolizes a sample from a multivariate Gaussian distribution of dimension equal to the number of observed datapoints x. θ represents a hyperparameter or lengthscale vector, and $K(\theta, x, x')$ symbolizes the covariance matrix between all possible pairs of (x, x') for a given set of hyperparameters.

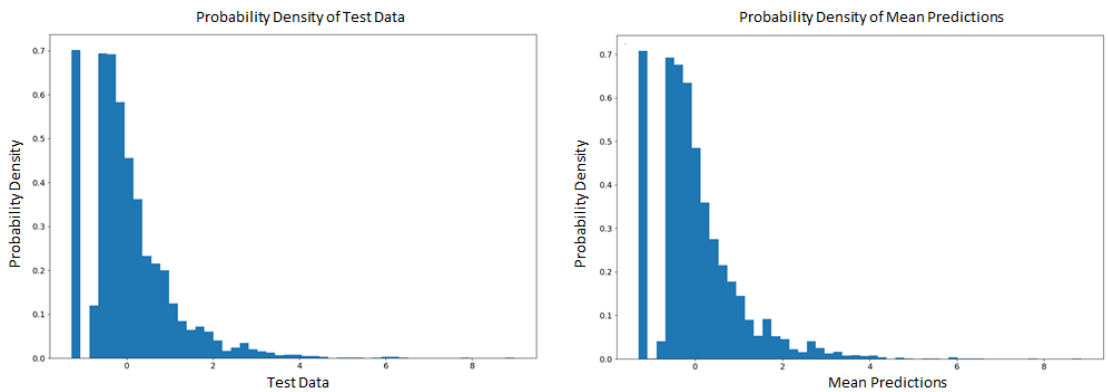

Figure 3.7. Probability densities of test turnaround duration and mean predictions of turnaround duration for delayed flight schedule instances in Weather domain

As previously mentioned, a subset of the data set defined only by flight schedules delayed by weather incidents is used for the *GPR* demonstration. This subset of data is split into two separate sets of training and test data respectively, such that the training data (70% of the data subset) is used to fit the *GPR* model for actual turnaround duration and the test (i.e. new or unseen) data is used to validate the model by verifying that the test data is consistent with mean predictions from the model. Plotting the probability density function of the test data, revealed a lognormal distribution of the actual turnaround duration in the data set, as evidenced by Fig.3.7. Hence, the Matern32 kernel function, which is

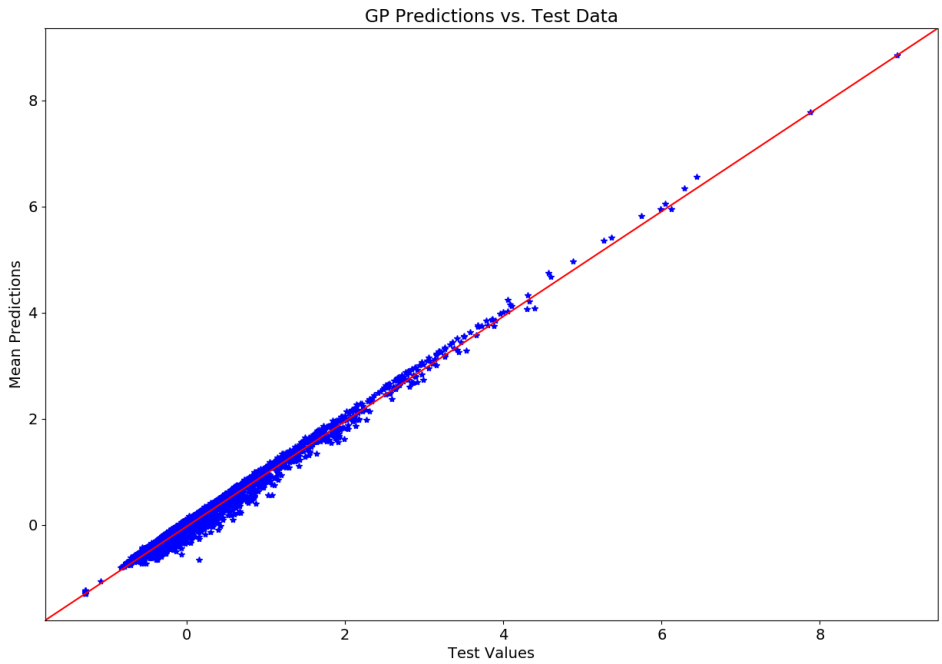

Figure 3.8. Mean predicted turnaround duration data vs. test turnaround duration data

a combination of Gamma and Bessel functions correlated by an hyperparameter of 3/2, is selected to fit the *GPR* model by means of a Gaussian process software named GPy [76].

Fig. 3.8 shows the plot of the mean predictions of the actual turnaround duration from the *GPR* model versus the actual turnaround duration from the test data, for which the turnaround duration values in both axes are scaled to a unit variance from the mean of the data values. The red diagonal line in Fig. 3.8 represents the 45-degree line, while each blue star represents a coordinate of the mean *GPR* prediction and test data describing actual turnaround duration for each datapoint (i.e. instance of weather-delayed flight schedule). Fig. 3.8 shows that the coordinates for the datapoints follow the trend of red diagonal line

Table 3.2. Lengthscales of refined data features for predicting actual turnaround duration through Gaussian process regression

GPR.Mat32.lengthscale	**Feature Class**	**Feature Name**
1738.68	Determinate Aleatoric	*sin_date*
1650.37	Determinate Aleatoric	*cos_date*
1857.16	Determinate Aleatoric	*orig_x_dir*
1808.49	Determinate Aleatoric	*orig_y_dir*
1652.50	Determinate Aleatoric	*orig_z_dir*
1750.13	Determinate Aleatoric	*ONBD_CT*
1193.69	Determinate Aleatoric	*SCHED_TURN_MINS*
44.77	Epistemic	*ADJST_TURN_MINS*
1367.11	Determinate Aleatoric	*schd_acft_type*
1367.11	Epistemic	*actl_acft_type*
1406.15	Epistemic	*SWAP_FLT_FLAG*
975.98	Indeterminate Aleatoric	*ATC Hold at Origin*
972.46	Indeterminate Aleatoric	*ATC Hold at Destination*
1.00	Indeterminate Aleatoric	*Deicing at Gate*
156.75	Indeterminate Aleatoric	*Ice on Wings*
1.00	Indeterminate Aleatoric	*Lightning Strike*
1.00	Indeterminate Aleatoric	*Turbulence*
1.00	Indeterminate Aleatoric	*Hail or Snow Damage*

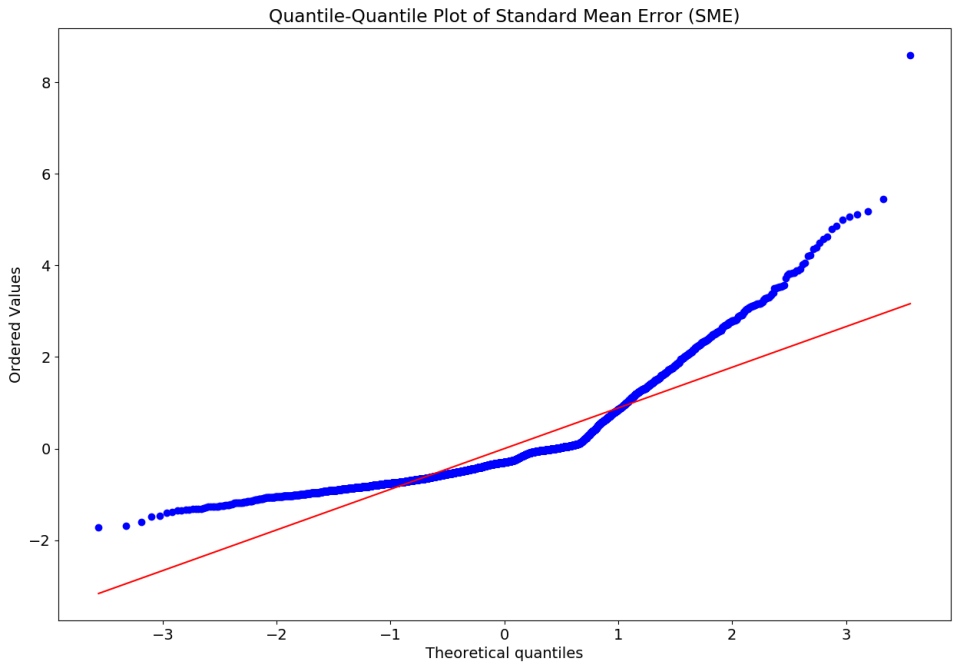

Figure 3.9. Quantile-Quantile plot of standard mean error between predicted and test data for actual turnaround duration

almost perfectly (root mean square error of 9%), which indicates that the *GPR* model is able to effectively predict "unknown" actual turnaround duration. Each coordinate that falls on the red line implies an exact prediction of the test data by the *GPR* model, and as such, Fig. 3.8 shows that the *GPR* model perfectly predicts actual turnaround periods that lie over six standard deviations away from the mean.

Table 3.2 shows the values of the optimized hyperparameters (i.e. length-scales) of refined flight schedule and disruption features for estimating actual turnaround time. Lower values in Table 3.2 indicate higher importance of features for predicting actual turnaround period. Similar to the result from *MIR*, turnaround duration adjusted during schedule execution (i.e. *ADJST_TURN_MINS*)

is the most significant epistemic flight schedule feature for predicting actual turnaround duration, as indicated by its low lengthscale value of approximately 45. Of all the aleatoric features (determinate and indeterminate), disruption features representing deicing at the gate, lightning strike, turbulence, and hail and snow damage (all with lengthscale values of 1) are the most significant for accurately estimating actual turnaround period during schedule execution.

Fig. 3.9 shows the quantile-quantile (QQ) plot of the standard mean error (SME) between the mean predictions from the *GPR* model and the test data for actual turnaround duration. The straight red line in Fig. 3.9 represents the trend line for a standard normal distribution, and as such, the bi-linear trend for standard mean error (portrayed by the blue dots) in Fig. 3.9 confirms that the distribution of actual turnaround period for weather-delayed flight schedules is lognormal. Categorically, the overlapping trend between the 45-degree red line and the spread of the coordinates in Fig. 3.8, coupled with the lognormal distribution trend noted from the QQ plot in Fig. 3.9, validates that the turnaround process for weather-delayed flight schedules is indeed a Gaussian process. To that effect, the relationship between the data features (shown in Table 3.2) and the actual turnaround duration during schedule execution (i.e. target feature) can be described by a Matern32 covariance function.

Chapter Summary: This chapter provided macroscopic and microscopic summaries of the historical airline scheduling and operations data necessary for creating high fidelity models for airline disruption management. Through macroscopic analysis, we identified that over 94% of the irregular operations over a one-year period (from a major U.S. airline) occurred due to different forms of flight schedule delays. To that end, we investigated crucial drivers and properties for effectively managing flight schedule delays through microscopic analysis of weather-delayed flight schedule data, which also demonstrated a toolbox for applying appropriate machine learning techniques to enable data-driven simultaneously-integrated recovery during disruption

management. In the next chapter, we extensively discuss the processes and routines used to obtain high fidelity data-driven models for simultaneously-integrated recovery.

4. CREATING INTELLIGENT AGENTS FOR A *SIR* PARADIGM IN AIRLINE DISRUPTION MANAGEMENT

Most airlines incorporated computer support for proactively and reactively managing their operations in the 1970s. However, that support is presently restricted to informing human managers in the AOCC about the resources available to their functional roles and domains, and what activities the resources are involved in at a particular time [47]. As such, the current practice for airline disruption management requires that actionable decisions are made by human managers (or specialists) in the AOCC exercising judgement. Human specialists are capable of readily identifying key factors that can affect the direction of an actionable decision for airline disruption management. However, human managers can not quickly evaluate copious amounts of information to make flexible and readily scalable decisions for managing irregular operations [24], and adding more personnel to the functional roles in the AOCC can not effectively offer nor increase the bandwidth necessary to address this limitation [26]. Hence, there is a need to develop suitable models (i.e. intelligent agents), for assisting human specialists, to expeditiously generate high quality decisions during disruption management and operations recovery.

Predictive analytics is an emerging advancement in traditional data analytics that aims to enable the creation of seamless platforms for making predictions about future outcomes based upon historical data and appropriate analytics techniques such as statistical modeling and machine learning [77]. To that effect, predictive analytics provides an appropriate medium to readily estimate decision-making components of intelligent domain managers at different instances during airline scheduling, so as to enable real-time disruption management. Unlike exploratory analysis that reveals inherent patterns in data, predictive analytics represents a set of business intelligence

technologies that uncovers forward-looking patterns within flight scheduling and operations data that can be used to predict human decision-making behavior at any instance in time during airline disruption management [78]. There are two primary categories of predictive analytics, based upon the manner in which data is mined to develop an intelligent agent, namely: *supervised learning* and *unsupervised learning*.

- *Supervised learning*: Supervised learning represents the process of utilizing historical data sets on airline operations that contains specific features of interest (i.e. features to be estimated) to develop intelligent agents that are capable of predicting future outcomes of the features of interest during disruption management. As such, supervised learning requires the appropriation of a subset of the data for training the intelligent agent, and another data subset for testing and validating the predictions (i.e performance) of the intelligent agent in new situations. Primary approaches for training intelligent agents (or models) through supervised learning include classification, regression, and time-series analysis. Classification methods distinguish the groups to which different entries in a data set belong based upon latent data properties. For instance, classification can be used to teach an intelligent agent how to identify whether or not a specific flight schedule arrived on time. Regression forecasts future outcomes from past outcomes by providing an analysis of variance between predicted and actual values of features of interest. Continuous values for flight schedule features such as block time, turnaround duration, and delay duration can be estimated through regression. Similar to regression, time series analysis predicts future outcomes of continuous values for temporal features by accounting for unique properties like periodicity of time and calendars. As such, the estimation of exact departure and arrival times in a day, by an intelligent agent, can be facilitated through time series analysis techniques.

- *Unsupervised learning*: Unlike supervised learning that primarily relies on a cache of results imbued in a trained model, unsupervised learning uses descrip-

tive statistics [79, 80] to assess the natural patterns and associations that occur among features within a data set and does not predict a value for a target feature [81]. To that effect, the main objective of unsupervised learning is to extract and model the underlying structure or distribution of a data set to define the behavioral properties of an intelligent agent. Unsupervised learning approaches can be categorized into clustering and association. Clustering is analogous to classification in supervised learning, but unlike classification, the grouping of data features into clusters is not defined by dependence on a target feature, but rather on algorithms defined by specific and implicit statistical rules. Association addresses problems where the objective is to discover the rules that describe the interactions between features for large portions of data. As such, unsupervised learning through association can be used to develop intelligent agents capable of predicting sequences of flight schedule features that represent decision actions during airline disruption management.

The existing integrated recovery paradigms for airline disruption management employ monolithic system design methods that rely on the development of specific rules and requirements before a system that meets the specifications is designed [47]. As such, current design methods are unable to readily accommodate additional system complexities resulting from the introduction of new capabilities to the system. Furthermore, existing systems that enable the current practice in airline operations recovery and disruption management are unable to effectively quantify uncertainty in decision-making at stable intermediate forms during disruption management [4, 27].

Thus, from an airline scheduling and disruption management perspective, predictive analytics presents an avenue to create intelligent domain managers for a simultaneously-integrated recovery paradigm by enabling studies that aim to actively allow and enhance modular and adaptive decision-making capabilities for intelligent domain managers [82, 83]. To that effect, this chapter extensively discuses the principles and routines used for learning and predicting long-term behavioral properties and short-term performance proclivities during airline disruption management, through

the development of two separate models that define an intelligent manager for a representative functional domain in the AOCC. Hence, the uncertainty and predictive transfer function models (i.e. UTFM and PTFM) are distinct components of an intelligent domain manager that define the domain manager's capacity to efficiently estimate its disruption resolution actions and performance impact, respectively, for a disrupted flight schedule.

4.1 Data Preprocessing

Prior to learning and assembling the UTFM and PTFM frameworks to enable the estimation of uncertainty propagation and resolution performance patterns during airline disruption management, it is imperative to define the nature of the airline data set that will be used to develop the constituent models for intelligent domain managers. By following the data nomenclature elucidated in Chapter 3, this section briefly introduces the methods used to abstract and encode data features in the historical scheduling and operations data set to achieve high-fidelity intelligent models.

Table 4.1 shows the data preprocessing methods applied to facilitate the development of intelligent agents for airline disruption management. As demonstrated in Chapter 3, many algorithms for learning artificial intelligence models perform best with continuous data [84]. Hence, it is necessary to encode all applicable feature values in the raw data set into functional and relevant continuous data for use in appropriate algorithms. Therefore, temporal data features defined by dates are transformed into periodic (i.e. sine and/or cosine) vectors based upon the season of the year, month of the year, day of the week and day of the year, partly based upon the seasonal behavior trends observed from principal component analysis on a data subset of weather-related disruptions from prior exploratory data analysis in Chapter 3. In that regard, data features representing geographical locations such as departure and arrival stations are first transformed into spherical directional vectors based upon the longitude and latitude coordinates of their corresponding airports, and subsequently

Table 4.1. Data preprocessing for development of intelligent agents for disruption management

Raw Data Class	***First-Degree* Transformation**	***Second-Degree* Transformation**	**Refined Data Type**
Geographic Features	Spherical directional vectors, geodesic distance	Standardization	Continuous
Temporal Features	Periodic (Sine/Cosine) vectors	Standardization	Continuous
Categorical Features	One-hot encoding	Standardization	Continuous
Continuous Features	N/A	Standardization	Continuous

transformed into the distance between the departure and arrival airports on an oblate spheroid Earth via the Vincenty geodesic equation [56]. Temporal time-of-day events (e.g. departure and arrival times) are transformed into two separate periodic vectors of different amplitudes, based upon a 24-hr clock period and the percentage of 8-hr work shift completed (at the time of departure or arrival) by human specialists in the AOCC, respectively. The work shift characterization of time-of-day events, via a periodic vector, is intended to capture and represent daily disruption resolution proclivities of human specialists, which can be induced by how much time the specialists have to address a disruption before their work shift is complete. Categorical features in the data set are transformed into sparse matrices through one-hot encoding [57], and all data features (fields) in the data set are subsequently scaled to obtain a standard normal distribution (i.e. standardization) to facilitate statistical interpretation

and assessment of results obtained from the intelligent models. A complete definition of all the refined data features used for creating the intelligent agents discussed in this dissertation can be found in the Appendix A.

The subsequent sections in this chapter provide extensive discussions of the tenets and techniques employed to develop the uncertainty transfer function model and predictive transfer function model for an intelligent domain manager (or specialist agent) in the AOCC.

4.2 Uncertainty Transfer Function Model (UTFM)

From a statistical perspective, the main objective of disruption management is to eradicate the functional impact of aleatoric uncertainty [85] that stems from random occurrence of disruptive events like inclement weather (i.e. uncertainty from indeterminate aleatoric data features) on optimal schedule execution on the day of operation. However, the state of the art for attaining the primary objective of airline disruption management introduces epistemic uncertainty in resolving disruptions at each phase of flight when human specialists, with different experience levels and perspectives, are required to make decisions that will affect the disruption resolution action implemented in a subsequent flight phase. Although existing approaches for airline disruption management are capable of mitigating the effect of uncertainty from indeterminate aleatoric features during scheduled flight operations, they are limited by the incapacity to explicitly address epistemic uncertainty and its impact on the quality of resolutions applied for schedule recovery and disruption management. Advancements in unsupervised learning techniques for big data [30, 31, 86], coupled with cost-efficient computational data storage platforms [87], have presented an avenue for the development and assessment of predictive and prescriptive models to facilitate the exploration of new approaches for addressing uncertainty during airline disruption management.

[9] introduced and demonstrated the first and only published application of principles from unsupervised learning in airline disruption management that enables simultaneously-integrated recovery of all problem dimensions. Although [9] provide a qualitative and quantitative framework for discerning and modeling adaptive decision-making for airline disruption management, their approach is statistically inefficient because the model-free environment, wherein intelligent agents interact through reinforcement learning [43], does not employ (nor estimate) a predefined flight schedule and operations model consistent with airline scheduling practices to obtain optimal disruption resolutions during airline schedule recovery. As a result, their approach requires considerable trial-and-error experience to obtain acceptable estimates of future consequences from adopting specific disruption resolutions during airline disruption management. In contrast with the work by [9], our proposed uncertainty transfer function model (or UTFM) framework leverages real-world historical data to eliminate the necessity of trial-and-error experience for facilitating simultaneously-integrated recovery during airline disruption management. Essentially, the UTFM offers a robust approach that utilizes historical airline data on different *rules-of-thumb* employed by human specialists in the AOCC, together with current practices in airline schedule operations and recovery, to effectively quantify and minimize the propagation of epistemic uncertainty (i.e. uncertainty stemming from epistemic data features) in decision-making during disruption management.

4.2.1 UTFM Methodology

The debilitating effect of disruptions on the optimal execution of a scheduled revenue flight becomes more pronounced with increasing number of flight legs [26]. According to the International Air Transport Association (IATA), a scheduled revenue flight is any flight schedule executed by an airline for commercial remuneration according to a published timetable, and each flight leg in a scheduled revenue flight

represents an aircraft's trip from one airport to another airport without any intermediate stops.

Figure 4.1. Disruption management for a scheduled flight defined by a Markov decision process

Every flight leg in a scheduled flight is defined by phases of aircraft activity (or flight phases) that are influenced by the decision-making activities of multiple air transportation stakeholders as the aircraft journeys between airports. For an airline, human specialists located in the AOCC perform important decision-making activities at respective flight phases during each flight leg in a scheduled flight, where actions implemented during the most precedent flight leg influence the changes in schedule and decisions made in subsequent flight legs. Thus, fundamentally, the decision-making process for managing disruptions in a scheduled flight adheres to the Markov property [88], as illustrated in Fig. 4.1. Congruently, schedule changes and decisions at a future flight phase (conditional on both past and present flight phases) during a flight leg are strictly dependent on the schedule changes and decisions made for mitigating irregular operations in the present flight phase, and not on the sequence of schedule changes and decisions made during the flight phases that preceded it.

Problem Formulation as a Relational Dynamic Bayesian Network

We formulate our UTFM framework for airline disruption management as a relational dynamic Bayesian network (RDBN) [84, 89, 90] wherein the modeling domain is defined as an airline route network containing multiple related flight schedules that are routinely executed and randomly disrupted over a certain time frame. The RDBN architecture provides a generative modeling approach that defines a probability distribution over instances of scheduled (disrupted) flights in an airline route network. By

employing data features (attributes) that provide a logical description of airline activities for disruption management coupled with probabilistic graphical model templates (schema), the RDBN architecture defines the probabilistic dependencies in a domain across two time slices. Thus, for our RDBN architecture, the following general and interrelated definitions apply [86, 90, 91]:

Definition 1 (*Dynamic Relational Domain*)

Syntax: *A term represents any flight phase, flight leg, or flight schedule in an airline route network domain. A predicate represents any concatenation of attributes or activities for any term in the domain.*

- *The dynamic relational domain is the set of constants, variables, functions, terms, predicates and atomic formulas* $Q(r_1, ..., r_n, t)$ *that define an airline route network, such that each argument* r_i *is a term and* t *is the time step during disruption management.*

- *The set of all possible ground predicates at time* t *is determined by substituting the variables in a low-level schema of each argument with constants and substituting the functions in a high-level schema of each argument with resulting constants.*

Semantics: *The state of an airline route network domain at time* t *during disruption management is the set of ground predicates that are most likely at time* t.

Assumptions:

- *The dependencies in an airline route network domain are first-order Markov such that ground predicates at time* t *can only depend on the ground predicates at time* t *or* $t-1$.

- *A grounding (i.e. referential learning or decoding process) in an airline route network domain at time* $t-1$ *precedes a grounding at time* t*, such that this assumption takes priority over the ordering between predicates in the domain.*

$$Q(r_1, ..., r_n, t) \prec Q(r'_1, ..., r'_m, t') \quad \textit{if} \quad t < t'$$

Definition 2 (*Two-time-slice relational dynamic Bayesian network: 2-TRDBN*)
Syntax: *The 2-TRDBN is any graph (or schema) that provides a probability distribution on the state of an airline route network domain at time $t+1$ given the state of the domain at time t.*
Semantics: *For any predicate Q bounded by groundings at time t, we have*:

- *A set of parents $Pa(Q) = \{Pa_1, ..., Pa_l\}$, such that each Pa_i is a predicate at time $t-1$ or t.*
- *A conditional probability model for $P(Q|Pa(Q))$, which is a first-order probability tree (or a trellis) on the parent predicates.*

Assumptions:

- *If Pa_i is at time t, then $Pa_i \prec Q$ or $Pa_i = Q$.*
- *If $Pa_i = Q$, then its groundings are bounded to those that precede the defined grounding of Q.*

Definition 3 (*Relational Dynamic Bayesian Network: RDBN*)
Syntax: *A RDBN for disruption management is any network pair $(\mathcal{N}_{\prime}, \mathcal{N}_{\rightarrow})$, such that $\mathcal{N}_{\prime}$ is a dynamic Bayesian network (DBN) at time $t = 0$ and $\mathcal{N}_{\rightarrow}$ is a 2-TRDBN.*
Semantics: *$\mathcal{N}_{\prime}$ characterizes the probability distribution over a relational (airline route network) domain prior to schedule execution (i.e. at $t = 0$). Given the state of the relational domain at a time t during disruption management (or schedule execution), $\mathcal{N}_{\rightarrow}$ represents the transition probability distribution on the state of the domain at time $t+1$.*
Assumptions: *A term (node) is created for every ground predicate and edges are added between a predicate and its parents at a time $t > 0$.*

- *Parents are obtained from $\mathcal{N}_{\prime}$ if $t = 0$, else from $\mathcal{N}_{\rightarrow}$.*
- *The conditional probability distribution for each term is defined by a probabilistic graphical model bounded by a specific grounding of the predicate.*

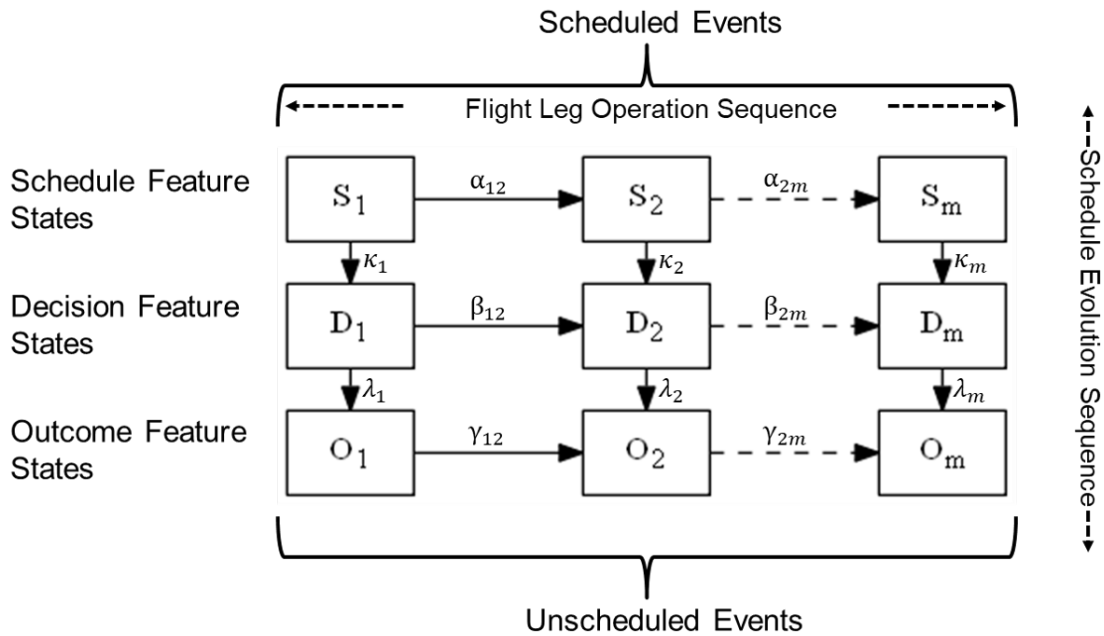

Figure 4.2. RDBN architecture for a representative flight leg

For the purposes of uncertainty quantification and propagation discussed in this Chapter, we adapt the aforementioned definitions for a RDBN to construct a UTFM, such that the modeling domain is for a representative flight leg that is defined by the probabilistic graphical model (i.e. atomic formula) illustrated by Fig. 4.2. The flight leg operation sequence (i.e. disruption progression along horizontal axis in Fig. 4.2) represents the spatiotemporal axis in a multidimensional Markov chain [92] that describes the order in which (or when) random disruptions (i.e. indeterminate aleatoric features such as bad weather events) occur during different phases of flight. As such, the flight leg operation sequence defines the propagation of aleatoric uncertainty in schedule and operations recovery. The schedule evolution sequence (i.e. schedule-planning evolution along vertical axis in Fig 4.2) captures epistemic uncertainty in decision-making for operations recovery by characterizing the order in which (or how) the flight schedule changes with respect to disruption resolutions such as *rules-of-thumb* or decision features like delay period applied by human specialists

on the day of operation. Scheduled events constitute data features (such as departure times, arrival times, aircraft type, etc.) that define the optimal airline (flight) schedule for m different flight phases prior to schedule execution.

Furthermore, scheduled events serve as start points in the UTFM architecture and may also inform the decision-making of human specialists during the resolution of a specific type of disruption. Unscheduled events represent an updated set of data features that characterize the adjustment of optimal flight schedule by human specialists based upon the impact of disruption at m different flight phases during schedule execution. Unscheduled events provide end points in the UTFM architecture. Schedule feature states (labeled S in Fig. 4.2) represent functions of data items that are strictly subject to uncertainty in determinate aleatoric data features with respect to airline planning and scheduling prior to schedule execution. Decision feature states (labeled D in Fig. 4.2) represent functions of action items that human specialists implement during schedule execution to resolve disruptions in the optimal schedule obtained prior to schedule execution (e.g. delay time, flight swap flag, etc.), while outcome feature states (labeled O in Fig. 4.2) represent functions of data items that human specialists use to assess the impact of their decisions after resolving deviations from the optimal schedule obtained prior to schedule execution. The parameters for $S, D, O, \alpha, \beta, \gamma, \kappa, \lambda$ in Fig. 4.2 are obtained by grounding (algorithms in Appendix B) via hidden Markov models, to determine the schedule evolution and decision-making proclivities of human specialists at each flight phase during disruption management for a characteristic flight leg.

4.2.2 Solution Approach for UTFM

We use a solution technique based upon a component assembly process, which enables generative programming for probabilistic graphical models [86], to calibrate (ground) the parameters of the multidimensional Markov chain that define the UTFM introduced in Section 4.2.1. Component assembly is a widely espoused modeling

paradigm in computer science and software engineering [93], and facilitates the integration of state components of the UTFM that define separate phases of flight operation and schedule-recovery evolution in the UTFM architecture. Through generative programming [94, 95], highly customized and optimized intermediate parameters defining each state component and aggregate UTFM parameters, can be created on demand from elementary and reusable parameters of state components, through a priori knowledge of the graph structure of the Markov system.

Fig. 4.3 reveals our solution approach to automatic uncertainty quantification for airline disruption management. The approach starts by abstracting historical

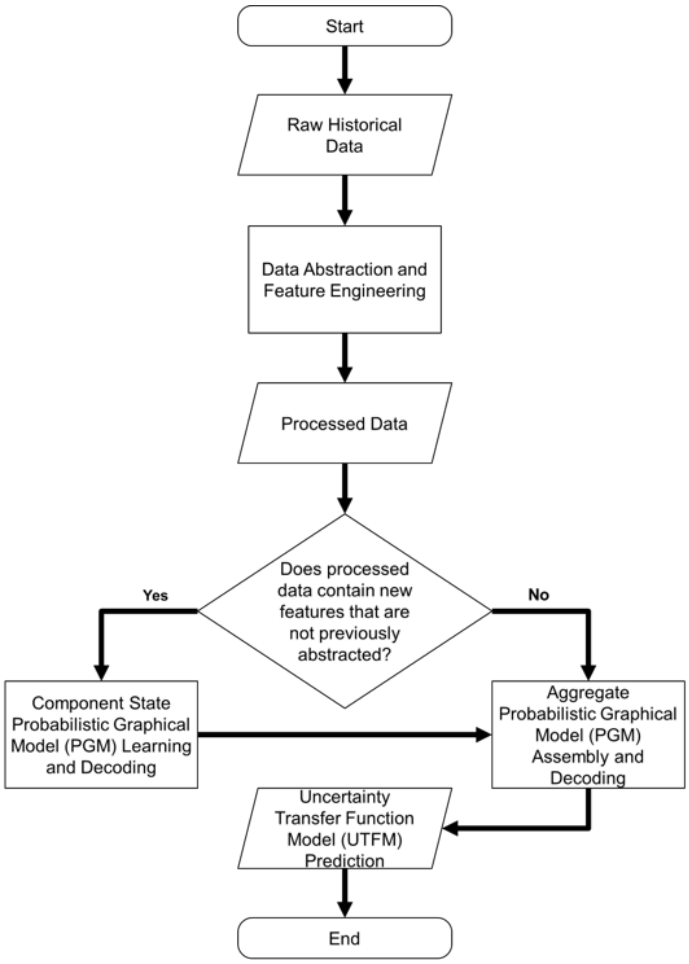

Figure 4.3. Component assembly approach for automatic uncertainty quantification for disruption management

airline schedule and operations recovery data into a digestible data set, appropriate for machine learning algorithms [51,96]. Next, the refined data set is used to learn optimal probabilistic graphical model parameters of each state component of the UTFM, before constructing an overarching probabilistic graphical model from the aggregation of the respective optimized probabilistic graphical models of state components.

For the remainder of this section, we introduce probabilistic graphical modeling and discuss the role of hidden Markov models for grounding (i.e. calibrating the parameters) in a probabilistic graphical model representation of the UTFM.

Probabilistic Graphical Modeling

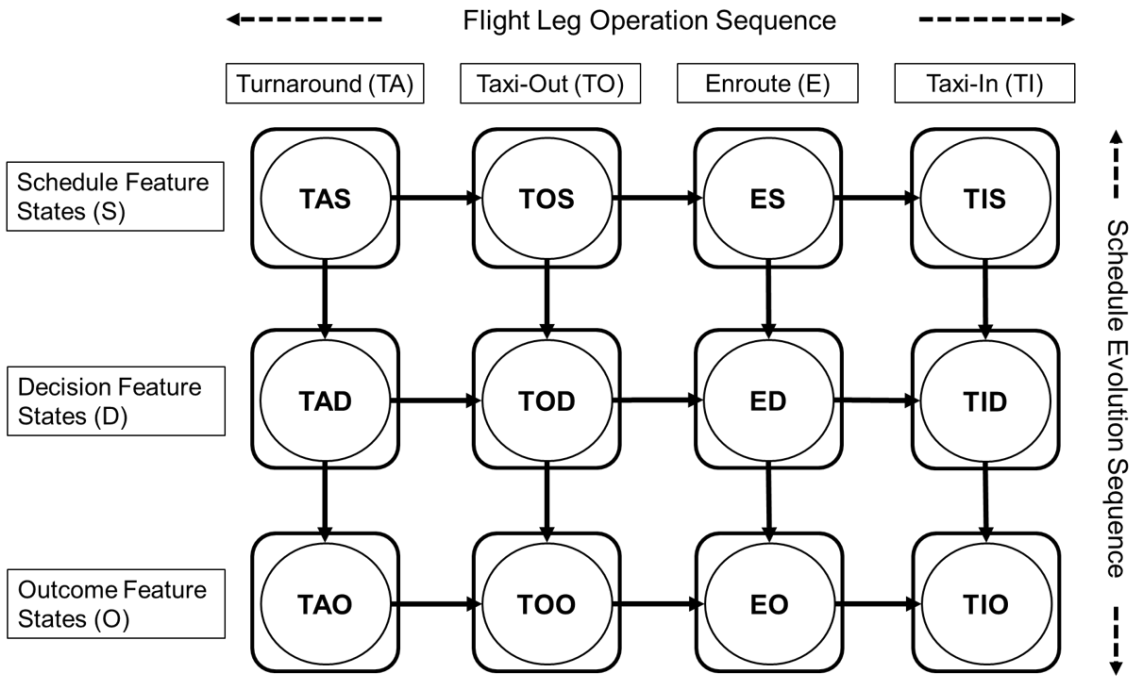

Figure 4.4. Probabilistic graphical model representation of UTFM

Probabilistic graphical modeling provides an avenue for a data-driven approach to constructing the UTFM architecture, which is very effective in practice [86]. By employing rudimentary activity guidelines from human specialists in the AOCC for

airline disruption management, critical components for constructing an intelligent system such as representation, learning, and inference can be readily inculcated in the UTFM.

Fig. 4.4 shows the probabilistic graphical model representation of the UTFM defined by four major phases of flight along the operation sequence axis namely: Turnaround, Taxi-Out, Enroute, and Taxi-In, while the schedule evolution sequence axis is defined by three separate phases of schedule changes with respect to airline planning on day of operation namely: Schedule, Decision, and Outcome. Thus, the graph structure of the UTFM comprises of 12 distinct component states (nodes) with 12 internal state transitions and 17 external state transitions, such that each component state contains a set of combination (interaction) of data features, listed in Section 4.2.3, that encode the behavioral proclivities of human specialists at different phases of activity during airline disruption management.

Schedule state components (i.e., TAS, TOS, ES, TIS) in Fig. 4.4 represent an interaction of data features that describe the evolution of original (optimal) flight schedule predetermined prior to schedule execution on day of operation, which would inform the decision-making of a human specialist in the AOCC during schedule execution. As such, schedule state components in the UTFM encapsulate epistemic uncertainty in proactive disruption management prior to schedule execution (i.e., uncertainty in tactical disruption management). Decision state components in the UTFM (i.e., TAD, TOD, ED, TID) define the interaction of data features that describe the action items that human specialists implement for resolving specific types of disruption that occur during schedule execution, and define epistemic uncertainty in reactive disruption management during rescheduling on day of operation (i.e., uncertainty in operational disruption management). Outcome state components in Fig. 4.4 (i.e., TAO, TOO, EO, TIO) represent the interaction of a set of data features that characterize the original schedule adjusted based upon the impact of disruption resolutions (i.e. action items) implemented by human specialists during schedule execution, and therefore define epistemic uncertainty in proactive disruption management for future

airline scheduling after schedule execution (i.e., uncertainty in strategic disruption management).

Hidden Markov Models for Probabilistic Graphical Modeling of UTFM

The hidden Markov model (HMM), also known as a transducer-style probabilistic finite state machine [97], is the simplest class of dynamic Bayesian networks and a useful tool for representing probability distributions over a sequence of observations [98, 99]. The hidden Markov model obtains its name from defining two separate but related characteristics. First, it assumes that the observation at a particular instance in time was generated by an arbitrary process whose state is hidden from the observer. Second, it assumes that the state of this hidden process satisfies the Markov property. To that effect, the hidden Markov model lends an appropriate grounding medium for solving the learning and inference (decoding) problems [100] for the probabilistic graphical model representation and construction of the UTFM.

Mathematically, the hidden Markov model is defined as a stochastic process $(X_k, Y_k)_{k\geq 0}$ on the product state space $(E \times F, \mathcal{E} \otimes \mathcal{F})$ if there exist transition kernels $P : E \times \mathcal{E} \to [0, 1]$ and $\Phi : E \times \mathcal{F} \to [0, 1]$ such that

$$\mathbf{E}(g(X_{k+1}, Y_{k+1})|X_0, Y_0, ..., X_k, Y_k) = \int g(x, y)\Phi(x, dy)P(X_k, dx) \tag{4.1}$$

and a probability measure μ on E wherein

$$\mathbf{E}(g(X_0, Y_0)) = \int g(x, y)\Phi(x, dy)\mu(dx) \tag{4.2}$$

for any bounded and measurable function $g : E \times F \to \mathbb{R}$. As such, μ represents the initial measure, P is the transition kernel, and Φ represents the observation kernel of the hidden Markov model $(X_k, Y_k)_{k\geq 0}$.

HMM Learning

The learning problem for construction of the UTFM is representative of optimizing the parameters of the pair of dynamic Bayesian networks $(\mathcal{N}_{\prime}, \mathcal{N}_{\rightarrow})$ defined in Section 4.2.1 based upon available data, and therefore presents two separate learning sub-problems: *Intra-State* HMM learning and *Inter-State* HMM learning. Hence, *Intra-State* HMM learning and *Inter-State* HMM learning characterize the grounding process for obtaining optimal parameters for $\mathcal{N}_{\prime}$ and $\mathcal{N}_{\rightarrow}$ respectively. Specifically, *Intra-State* HMM learning represents the ability to effectively determine appropriate interaction patterns (i.e. transition likelihood) for *hidden* data features (subject to epistemic uncertainty) which are embedded in each state component of the UTFM shown in Fig. 4.4, based upon observing data features (i.e. observations) that are strictly subject to uncertainty from determinate or indeterminate aleatoric features observed at any phase of activity during airline disruption management. Some examples of data features that represent observations for *Intra-State* HMM learning of state components in the UTFM include total distance between origin airport and destination airport, and total number of passengers (i.e. demand for air travel) available for flight before and after schedule execution. Thus, the primary objective of *Intra-State* HMM learning is to achieve an optimal HMM (probability distribution mixture model) that is capable of efficiently predicting the likelihood of remaining at a particular phase of activity (i.e. state component) in the UTFM for airline disruption management.

Inter-State HMM learning, on the other hand, characterizes the ability to ascertain the interaction or transition patterns between any two neighboring state components (phases of activity) in the UTFM, wherein data features (listed in Section 4.2.3) embedded in the state component at the future (posterior) phase of activity in the UTFM are set as observations while data features embedded in the state component at the current (prior) phase of activity are set as hidden states. As such, the primary objective of *Inter-State* HMM learning is to attain an optimal HMM (probability

distribution mixture model) that is capable of accurately predicting the likelihood of transitioning between present and future phases of activity (i.e. state components) in the UTFM.

1. Compute

$$Q(\theta, \theta') = \sum_{z=\bar{Z}} log[P(X, z; \theta)]P(z|X; \theta') \tag{4.3}$$

2. Set

$$\theta'^{+1} = \arg\max_{\theta} Q(\theta, \theta') \tag{4.4}$$

The Baum-Welch algorithm [101] is a dynamic programming approach that uses the expectation maximization (EM) algorithm [102] to find the maximum likelihood estimate of the parameters of an HMM given a set of observations. The Baum-Welch algorithm presents a convenient means for learning the optimal parameters (i.e. state transition and emission probabilities) of an *Intra-State* or *Inter-State* HMM, because it guarantees that the optimal parameters of the HMM are easily estimated in an unsupervised manner during training by utilizing unannotated observation data [103]. In essence, the Baum-Welch algorithm described by steps in Equations 4.3 and 4.4,

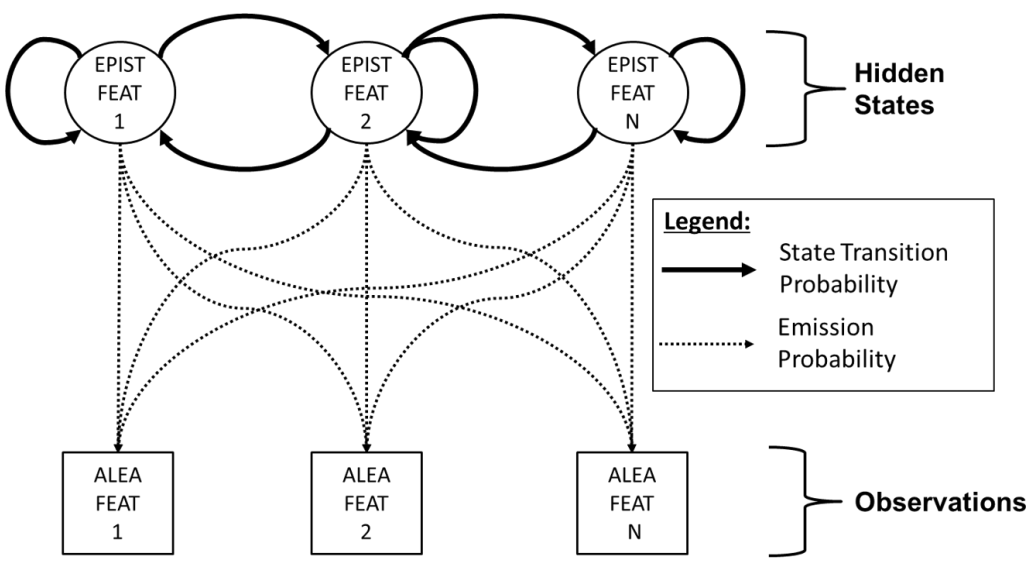

Figure 4.5. Intra-state HMM schema for remaining in an activity phase in UTFM

where X, $\bar{Z}$, and θ are the latent state space, observation space, and initial HMM parameters respectively, is an iterative procedure for estimating θ' until convergence, such that each iteration of the algorithm is guaranteed to increase the log-likelihood of the data. However, convergence to a global optimal solution is not necessarily guaranteed [101].

Fig. 4.5 reveals the general schema for learning the optimal parameters of an *Intra-State* HMM. The circles and squares in Fig. 4.5 represent the hidden (latent) states (i.e. data features subject to epistemic uncertainty) and observations (i.e. data features which are representative of aleatoric uncertainty) respectively. The learning objective for the *Intra-State* HMM schema in Fig. 4.5 is to use the Baum-Welch algorithm to find the optimal HMM parameters, which are the solid and dashed arrows that represent state transition probabilities and emission probabilities respectively.

Fig. 4.6 shows a generic schema for learning the optimal parameters of a typical *Inter-State* HMM, essential for predicting the likelihood of transitioning from one activity phase to another activity phase across both spatiotemporal axes in the UTFM. The circles labeled 'EPIST FEAT A' and 'EPIST FEAT B' in the *Inter-State* HMM

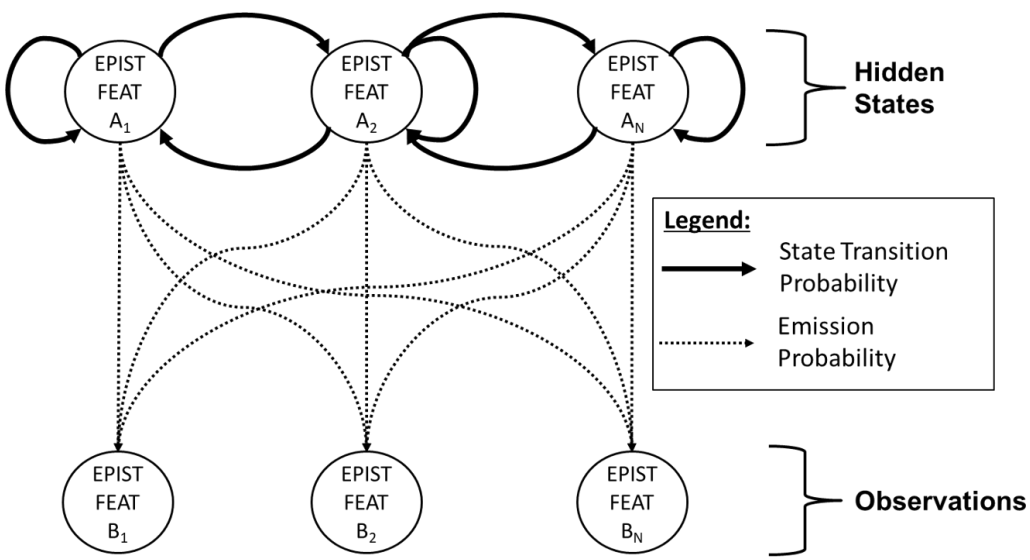

Figure 4.6. Inter-state HMM schema for transitioning between activity phases in UTFM

schema, shown in Fig. 4.6, represent epistemic data features embedded in current activity phase A (i.e. hidden states) and future activity phase B (i.e. observations), respectively, in the UTFM. Similar to the *Intra-State* HMM, the learning objective for the *Inter-State* HMM schema depicted by Fig. 4.6 is to use the Baum-Welch algorithm to find the optimal HMM parameters, which are the solid and dashed arrows that represent the state transition probabilities and emission probabilities respectively.

Unlike the *Intra-State* HMM schema where hidden states represent data features subject to epistemic uncertainty for disruption management and observations represent data features subject to aleatoric uncertainty, both hidden states and observations in the *Inter-State* HMM schema are representative of data features subject to epistemic uncertainty for disruption management. Thus, the overarching objective of an optimal *Intra-State* HMM is to accurately and expeditiously quantify the epistemic uncertainty at a specific phase of activity in the UTFM, while the overall objective of an optimal *Inter-State* HMM is to precisely predict the propagation of epistemic uncertainty between different phases of activity in the UTFM, for robust airline disruption management.

HMM Inference

Upon learning optimal parameters of *Intra-State* and *Inter-State* hidden Markov models, which define proactive and reactive behavioral patterns of human specialists at different stages of airline disruption management in the UTFM, it is imperative to conduct inference on the models to complete the assembly of the UTFM for effectively predicting uncertainty propagation patterns for airline disruption management. Similar to the learning problem, the inference problem for the assemblage of the UTFM is defined by two separate sub-problems: *component* UTFM decoding and *aggregate* UTFM decoding. *Component* UTFM decoding, defines the capacity of both *Intra-State* and *Inter-State* hidden Markov models for obtaining the most probable sequence of hidden (epistemic) data features in both types of HMMs, based upon (aleatoric or

epistemic) observation data features necessary for decoding in their respective schema illustrated in Figs. 4.5 and 4.6. Thus, the primary objective of *component* UTFM decoding problem is to provide the maximum likelihood estimates of the most probable sequence of hidden data features from optimal *Intra-State* and *Inter-State* HMMs upon inputting appropriate observation data features.

Aggregate UTFM decoding, on the other hand, describes the ability of the amalgamation of all *Intra-State* and *Inter-State* HMMs that constitute the UTFM, to precisely estimate the quantification and propagation of epistemic uncertainty at all phases of activity in the UTFM, based upon observing the maximum likelihood estimates of the most probable sequence of hidden data features retrieved from optimal *Intra-State* HMMs in the UTFM by way of *component* UTFM decoding. As such, a complementary objective of *aggregate* UTFM decoding problem is to obtain the parameters for $S, D, O, \alpha, \beta, \gamma, \kappa, \lambda$ as shown in Fig. 4.2, by estimating the weighted average of the maximum likelihood estimates of the most probable sequence of hidden data features retrieved from all optimal *Intra-State* and *Inter-State* HMMs upon observing their respective input data features (i.e. observations).

$$x^* = \arg\max_x P(z, x|\theta') \tag{4.5}$$

The Viterbi decoding algorithm [104, 105] is a proven dynamic programming algorithm that performs the HMM inference of the most probable sequence of hidden states (and its corresponding likelihood) based upon a specific sequence of observations, ultimately solving both the *component* and *aggregate* UTFM decoding sub-problems respectively. In principle, the Viterbi decoding algorithm defined by Equation 4.5, where x, z, and θ' represent sequence of hidden states, sequence of observations, and an arbitrary HMM respectively, uses a recursive (backtracking search) procedure for obtaining the optimal sequence of hidden states from the total number of possible sequences of hidden states for a specific sequence of observations, by selecting the sequence of hidden states that has the highest probability based upon maximum likelihood estimations from the arbitrary HMM [105]. As such, the Viterbi

decoding algorithm provides an efficient method for avoiding the explicit enumeration of all possible combinations of sequences of hidden states (i.e. concatenations of data features) while identifying the optimal sequence (i.e. Viterbi path) of hidden states with the highest probability of occurrence or least uncertainty [106].

In summary, from a UTFM assemblage perspective, the underlying objective of *component* UTFM decoding is to perform inference on all optimal *Intra-State* and *Inter-State* HMMs that define the UTFM, by implementing the Viterbi decoding algorithm to effectively estimate the likelihood (Viterbi probability) of the most likely sequence of hidden states (data features) based upon observing appropriate data features (observations), as shown in Figs. 4.5 and 4.6. By extension, the overall objective of *aggregate* UTFM decoding is to apply the Viterbi algorithm for determining the most likely sequence of state components that describes the propagation of epistemic uncertainty at different phases of activity in the UTFM shown in Fig. 4.4. The state transition parameters of a representative probabilistic finite state machine for the UTFM are weighted averages of the Viterbi probabilities obtained via *component* UTFM decoding that satisfy the properties of a stochastic matrix [107].

4.2.3 Computational Setup and Analysis

We now discuss the computational framework for generating state components of the probabilistic graphical model representation of the UTFM (shown in Fig. 4.4), which is used to predict epistemic uncertainty propagation during decision-making for airline disruption management. Prior to implementing the Baum-Welch and Viterbi algorithms to learn and decode useful HMMs for determining authentic likelihoods of internal and external transitions amongst different state components in the UTFM, raw historical airline data, necessary for enabling the application of algorithms for the development of these probabilistic graphical models, is first refined by following the data abstraction and feature engineering guidelines described in Section 4.1. Following data pre-processing and refinement, models are subsequently implemented

through learning and decoding in the Python programming language and facilitated by *pomegranate* [108], by utilizing a 56-core workstation running at 2.60 GHz with 192 GB of RAM.

Table 4.2.: List of features for *Intra-State* HMMs in UTFM.

***Intra-State* HMM for UTFM**	**Hidden States (Latent Data Features)**	**Observations (Observed Data Features)**
TAS (Turnaround Schedule)	*SWAP_FLT_FLAG, SCHED_ACFT_TYPE, SCHED_TURN_MINS, tod_sched_PB*	RTE, FREQ, PAX DMD
TOS (Taxi-out Schedule)	*taxi_out, tod_actl_TO, sched_block_mins*	RTE, FREQ, PAX DMD
ES (Enroute Schedule)	*actl_enroute_mins, tod_actl_LD, sched_block_mins*	RTE, FREQ, PAX DMD
TIS (Taxi-in Schedule)	*taxi_in, tod_sched_GP, sched_block_mins*	RTE, FREQ, PAX DMD
TAD (Turnaround Decision)	*shiftper_sched_PB, ADJST_TURN_MINS, DELY_MIN, SWAP_FLT_FLAG*	ORIG, DEST, FREQ, PAX DMD, DISRP
TOD (Taxi-out Decision)	*late_out_vs_sched_mins, shiftper_actl_PB, DELY_MIN*	ORIG, DEST, FREQ, PAX DMD, DISRP
ED (Enroute Decision)	*shiftper_actl_TO, shiftper_actl_LD, DOT_DELAY_MINS*	ORIG, DEST, FREQ, PAX DMD, DISRP
TID (Taxi-in Decision)	*DOT_DELAY_MINS, shiftper_sched_GP, shiftper_actl_GP*	ORIG, DEST, FREQ, PAX DMD, DISRP

TAO (Turnaround Outcome)	*SWAP_FLT_FLAG, ACTL_ACFT_TYPE, ACTL_TURN_MINS, tod_actl_PB*	RTE, FREQ, PAX DMD
TOO (Taxi-out Outcome)	*taxi_out, tod_actl_TO, actl_block_mins*	RTE, FREQ, PAX DMD
EO (Enroute Outcome)	*actl_enroute_mins, tod_actl_LD, actl_block_mins*	RTE, FREQ, PAX DMD
TIO (Taxi-in Outcome)	*taxi_in, tod_actl_GP, actl_block_mins*	RTE, FREQ, PAX DMD

4.2.4 UTFM Input and Output Features

Table 4.3.: List of features for *Inter-State* HMMs in UTFM.

***Inter-State* HMM for UTFM**	**Hidden States**	**Observations**
TAS → TOS	*SWAP_FLT_FLAG, SCHED_ACFT_TYPE, SCHED_TURN_MINS, tod_sched_PB*	*taxi_out, tod_actl_TO, sched_block_mins*
TOS → ES	*taxi_out, tod_actl_TO, sched_block_mins*	*actl_enroute_mins, tod_actl_LD, sched_block_mins*
ES → TIS	*actl_enroute_mins, tod_actl_LD, sched_block_mins*	*taxi_in, tod_sched_GP, sched_block_mins*

TAD → TOD	*shiftper_sched_PB, ADJST_TURN_MINS, DELY_MIN, SWAP_FLT_FLAG*	*late_out_vs_sched_mins, shiftper_actl_PB, DELY_MIN*
TOD → ED	*late_out_vs_sched_mins, shiftper_actl_PB, DELY_MIN*	shiftper_actl_TO, shiftper_actl_LD, DOT_DELAY_MINS
ED → TID	*shiftper_actl_TO, shiftper_actl_LD, DOT_DELAY_MINS*	*DOT_DELAY_MINS, shiftper_sched_GP, shiftper_actl_GP*
TAO → TOO	*SWAP_FLT_FLAG, ACTL_ACFT_TYPE, ACTL_TURN_MINS, tod_actl_PB*	*taxi_out, tod_actl_TO, actl_block_mins*
TOO → EO	*taxi_out, tod_actl_TO, actl_block_mins*	actl_enroute_mins, tod_actl_LD, actl_block_mins
EO → TIO	*actl_enroute_mins, tod_actl_LD, actl_block_mins*	*taxi_in, tod_actl_GP, actl_block_mins*
TAS → TAD	*SWAP_FLT_FLAG, SCHED_ACFT_TYPE, SCHED_TURN_MINS, tod_sched_PB*	*shiftper_sched_PB, ADJST_TURN_MINS, DELY_MIN, SWAP_FLT_FLAG*
TOS → TOD	*taxi_out, tod_actl_TO, sched_block_mins*	*late_out_vs_sched_mins, shiftper_actl_PB, DELY_MIN*
ES → ED	*actl_enroute_mins, tod_actl_LD, sched_block_mins*	*shiftper_actl_TO, shiftper_actl_LD, DOT_DELAY_MINS*

TIS → TID	*taxi_in, tod_sched_GP, sched_block_mins*	*DOT_DELAY_MINS, shiftper_sched_GP, shiftper_actl_GP*
TAD → TAO	*shiftper_sched_PB, ADJST_TURN_MINS, DELY_MIN, SWAP_FLT_FLAG*	*SWAP_FLT_FLAG, ACTL_ACFT_TYPE, ACTL_TURN_MINS, tod_actl_PB*
TOD → TOO	*late_out_vs_sched_mins, shiftper_actl_PB, DELY_MIN*	*taxi_out, tod_actl_TO, actl_block_mins*
ED → EO	*shiftper_actl_TO, shiftper_actl_LD, DOT_DELAY_MINS*	*actl_enroute_mins, tod_actl_LD, actl_block_mins*
TID → TIO	*DOT_DELAY_MINS, shiftper_sched_GP, shiftper_actl_GP*	*taxi_in, tod_actl_GP, actl_block_mins*

Table 4.2 and Table 4.3 reveal the hidden states (latent output features) and observations (observed input features) for all *Intra-State* and *Inter-State* HMMs, respectively, that constitute an aggregate HMM which defines the UTFM. The selection of specific hidden and observation data features, for all *Intra-State* and *Inter-State* HMMs that define the UTFM, was informed partly by literature [109–111], exploratory data analysis discussed in Chapter 3, and partly by discussions with human experts at the AOCC of the US airline that provided the raw historical data. We adopted this hybrid feature selection approach to ensure that data features which are appropriately relevant at a specific phase of activity in the UTFM are parameters of the corresponding HMM that represents that phase of activity for airline schedule planning and disruption management.

For *Intra-State* HMMs listed in Table 4.2, observations (i.e. observed aleatoric data features) are defined by data features that are strictly subject to aleatoric uncertainty with respect to how often they are considered, by human specialists,

in order to attain optimal schedules during the airline scheduling process shown in Fig. 1.1. Therefore, observations for *Intra-State* HMMs, listed in Table 4.2, include data features that represent the following: origin airport location and flight origin (ORIG), destination airport location (DEST), flight operating period in a calendar year (FREQ), route distance between origin and destination airports (RTE), number of passengers available for flight (PAX DMD), and random disruption types such as inclement weather (DISRP). ORIG, DEST, FREQ, RTE, and PAX DMD represent determinate aleatoric features that are determined by the airline, which are subject to aleatoric uncertainty at all phases of activity in the UTFM. As such, these features are indicative of the uniqueness of a particular flight schedule with respect to the airline route network. DISRP represents indeterminate aleatoric features that are subject to uncertainty which can not be readily controlled by an airline, and thus represent pure aleatory in airline disruption management. Hidden states (i.e. epistemic output data features) for *Intra-State* HMMs represent data features that are strictly subject to epistemic uncertainty with respect to the concatenation (interaction) of latent data features with the highest probability of occurrence, which indicate the activity patterns of human specialists (i.e. decision-making) for attaining optimal schedules during the airline scheduling process.

For *Inter-State* HMMs listed in Table 4.3, observations (i.e. observed epistemic data features) represent data features that are strictly subject to epistemic uncertainty with respect to the Viterbi probability (i.e. probability of the most likely sequence of latent data features estimated by an *Intra-State* HMM) at an immediate future phase of activity in the UTFM, while hidden states (i.e. latent epistemic data features) represent data features whose concatenations are strictly subject to epistemic uncertainty with respect to the Viterbi probability estimated by a characteristic *Intra-State* HMM in the present phase of activity in the UTFM during airline schedule planning and disruption management.

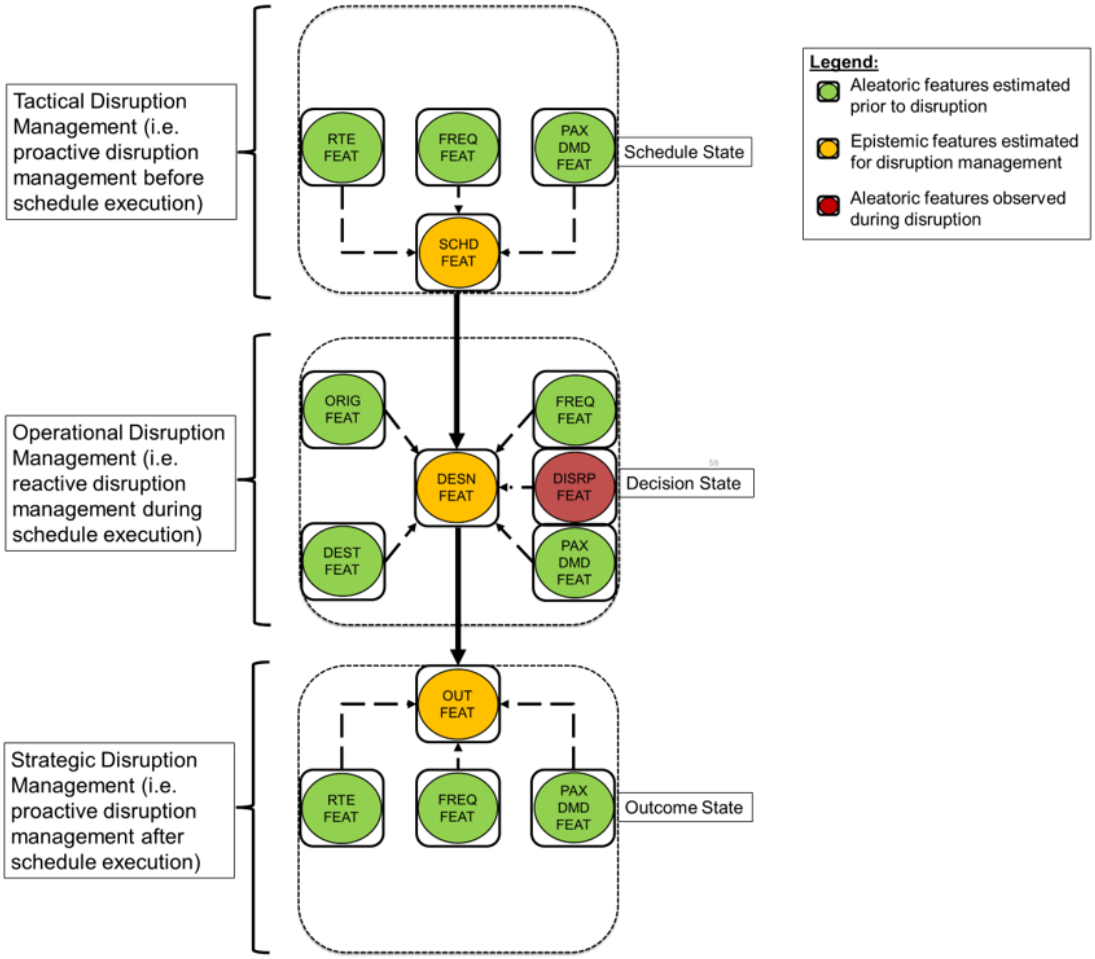

Figure 4.7. Phases of disruption management with respect to schedule execution

4.2.5 UTFM Learning

Defining Hidden States and Observations

Fig. 4.7 reveals a one-dimensional spatiotemporal representation of the UTFM reduced along the operation sequence axis (i.e. arbitrary column in Fig. 4.2). Yellow plates, indicated by SCHD FEAT, DESN FEAT, and OUT FEAT in Fig. 4.7, are representative of epistemic data features which define separate hidden states for *Intra-State* HMMs at each phase of flight operation along the operation sequence axis (i.e. Turnaround, Taxi-Out, Enroute, and Taxi-In) in the UTFM detailed in Fig. 4.4. In

that regard, SCHD FEAT represents data features that define hidden states for TAS, TOS, ES, and TIS *Intra-State* HMMs in the UTFM; DESN FEAT represents data features that define hidden states for TAD, TOD, ED, and TID *Intra-State* HMMs, while OUT FEAT is representative of data features that define hidden states for TAO, TOO, EO, and TIO states in the UTFM. Green and red plates in Fig. 4.7 are representative of uncertainty from determinate and indeterminate and aleatoric features for disruption management respectively, which define observations (inputs) for all *Intra-State* HMMs in the UTFM.

Data Segmentation for Learning

We employ the two separate lots of data in the full data set, defined in Chapter 3 as the *non-disrupted* and *disrupted* data sets, to learn optimal parameters of all HMMs that define different phases of activity for disruption management in the UTFM. The *non-disrupted* data set contains six hundred and twenty thousand instances of flight schedules in the airline network that executed as originally planned between September 2016 and September 2017. As such, the *non-disrupted* data set contains appropriate latent (hidden) and observation data features for flight schedules that executed without any uncertainty from indeterminate aleatoric features (i.e. random disruption features). Thus, we use the *non-disrupted* data set to calibrate *Intra-State* HMMs that define the tactical and strategic (i.e. Schedule and Outcome) phases of activity for disruption management in the UTFM. Unlike the *non-disrupted* data set, the *disrupted* data set contains all instances of flight schedules that executed through irregular operations due to delays in the airline route network from September 2016 to September 2017. Hence, the *disrupted* data set comprises of instances of flight schedules that executed with uncertainty from indeterminate aleatoric features over a one year period for separate functional roles in the AOCC. Therefore, we conduct *Intra-State* HMM learning for operational disruption management (i.e. Decision activity phases in the UTFM) by utilizing the *disrupted* data set. Similarly, we also utilize

the *disrupted* data set to learn the optimal parameters of all *Inter-State* HMMs along the operation sequence and schedule-change sequence axes in the UTFM for separate functional roles in the AOCC. To demonstrate the application of the UTFM in this Chapter, we only consider disruptions due to weather-related events (i.e. UTFM for the Weather functional role). As such, the *non-disrupted* data set is used to calibrate the *Intra-State* HMMs for tactical and strategic disruption management; a *disrupted* data set, with over twelve thousand instances of delayed flight schedules due to weather-related disruptions, is used to calibrate the *Intra-State* HMMs for operational disruption management and all *Inter-State* HMMs respectively. All *disrupted* and *non-disrupted* data sets used for training and validation are instantiated and segmented by using a random seed of 42 to ensure reproducible models.

Learning and Validation

Intra-State and *Inter-State* HMM learning for the development of the UTFM is implemented first by fitting data feature samples for hidden states to standard normal probability distributions that define the components of the initial measure of the UTFM. Next, samples (set) of observed data features are grouped as observations and the initial HMM state transition parameters are set as uniform distributions based upon the total number of hidden states, before invoking the Baum-Welch algorithm set to a convergence criterion of $1e^{-9}$. We perform a 5-fold cross validation [112] of Baum-Welch training on the sets of observations by examining marginal probability distributions of latent states across different folds to ensure modeling uniformity and generalizability, for approbation of a candidate optimal *Intra-State* or *Inter-State* HMM trained on the complete set of observations. The cross validation technique is used to assess the performance (goodness) of a trained HMM (for the UTFM) for estimating the likelihood of new observation (input) data, by verifying that the sums of the log likelihood of an appropriate test set of observations across each of the five folds and corresponding state probability distributions are consistent [113].

4.2.6 UTFM Decoding

Upon utilizing refined training data to learn the optimal parameters for *Intra-State* and *Inter-State* HMMs, a hidden Markov model (i.e. probabilistic finite state machine) representation of the UTFM is assembled to enable the decoding of new (unseen) data that represent disrupted flight schedules, by setting the weighted estimates of Viterbi probabilities estimated from all *Intra-State* and *Inter-State* HMMs as parameters of the aggregate *left-right* HMM that represents the UTFM, before applying the Viterbi algorithm to decode (predict) the most likely sequence of state components (i.e. phases of activity during airline disruption management) in the UTFM due to observed inputs from a typical disrupted flight schedule.

Intra-State HMM Decoding

Fig. 4.8 shows an optimal state transition graph for hidden state features from a trained *Intra-State* HMM for remaining in the Turnaround Decision (TAD) phase of activity in the UTFM. Based upon the graph shown in Fig. 4.8, a specialist agent

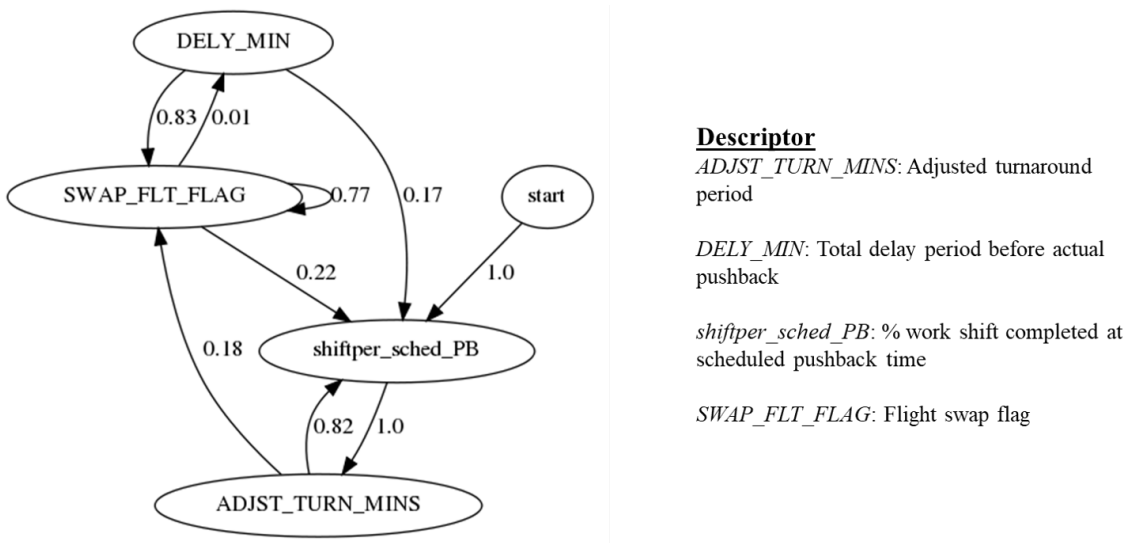

Figure 4.8. State transition graph of optimal *Intra-State* HMM for remaining in turnaround decision (TAD)

will commence decision-making for the turnaround phase of activity in the UTFM for operational disruption management first by assessing how much time there is until the scheduled aircraft pushback time, before considering (transitioning) to adjust the aircraft turnaround time (i.e. start probability of 1 and transition probability of 1). In the less likely event that the specialist agent does not return to assessing the time remaining prior to the scheduled aircraft pushback, a consideration to swap the aircraft is most likely (transition probability of 0.18) and there is a 77% likelihood that the process to swap the aircraft type will continue throughout the turnaround phase of flight operation during operational disruption management. Fig. 4.8 reveals that there is almost no prerogative for the specialist agent to consider delaying aircraft pushback time after swapping aircraft during the turnaround phase of flight operation for operational disruption management, as evidenced by the negligible transition probability of 1%.

Inter-State **HMM Decoding**

Fig. 4.9 shows an optimal state transition graph for hidden state features from a trained *Inter-State* HMM for transitioning from the Turnaround Decision (TAD) phase of activity to the Turnaround Outcome (TAO) phase of activity in the UTFM. From the graph shown in Fig. 4.9, a specialist agent will most likely commence the transitioning from operational decision-making for the turnaround phase of activity to strategic (proactive) decision-making for a future turnaround phase of activity in the UTFM for disruption management, first by assessing flight swap (start probability of 0.92 and internal state probability of 0.38), before a most likely transition to consider adjusting aircraft turnaround time (transition probability of 0.47 and internal state probability of 0.07). In the much less likely event that the specialist agent commences the transition to strategic disruption management by considering delay time before pushback first (start probability of 0.08 and internal state probability of 0.07), there is a 57% likelihood that the decision to adjust the turnaround time will follow, and

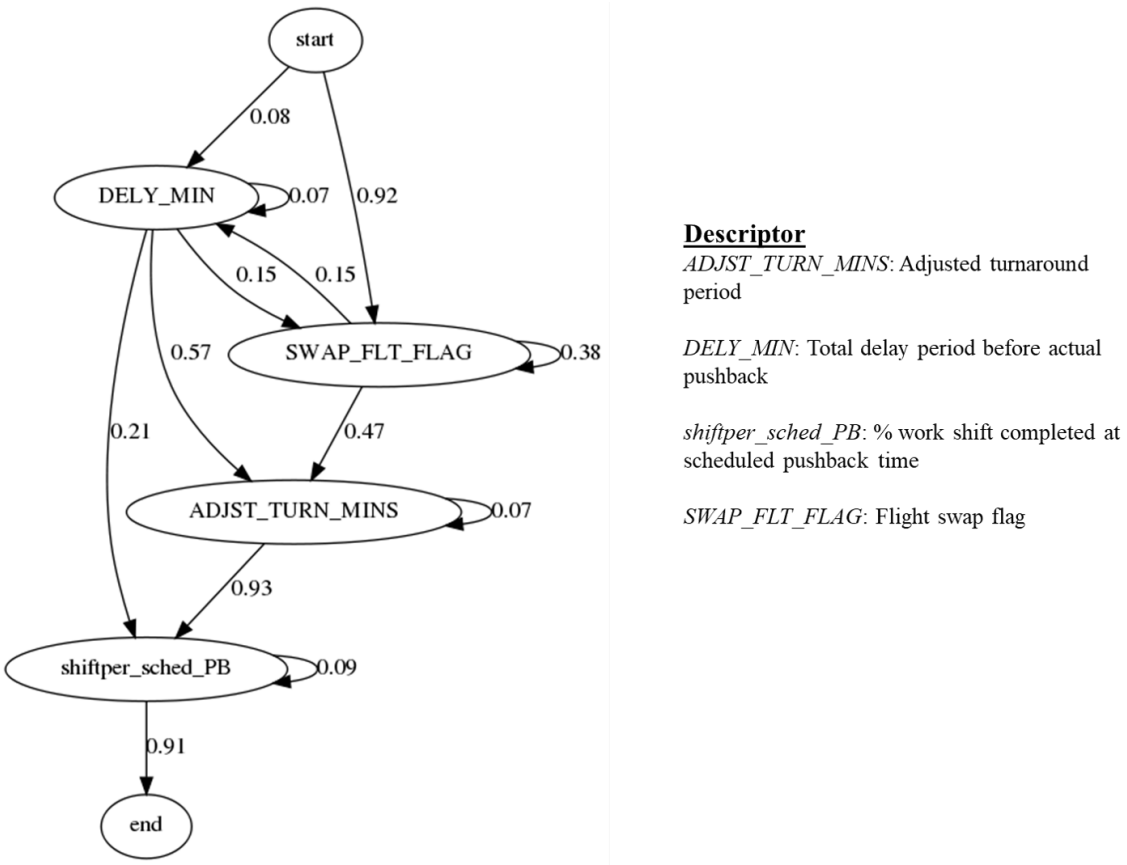

Figure 4.9. State transition graph of optimal *Inter-State* HMM for transition from turnaround decision (TAD) to turnaround outcome (TAO)

transitioning for strategic disruption management of the turnaround phase of future flight operation concludes by assessing the work shift (time available) for the next aircraft pushback schedule (end probability of 0.91 and internal state probability of 0.09).

Unlike the ergodic structure of the optimal state transition graph for the TAD *Intra-State* HMM represented in Fig. 4.8, the optimal state transition graph for the *Inter-State* HMM for transitioning between TAD and TAO phases of activity in the UTFM (depicted in Fig. 4.9) is modeled as a non-ergodic structure by introducing an absorption state (i.e. 'end' state) to characterize a definite transition process between both phases of activity. Thus, we apply ergodic (and non-ergodic) properties

to determine the optimal parameters of all *Intra-State* and *Inter-State* HMMs that constitute different phases of activity in the UTFM.

4.2.7 UTFM Results

We now evaluate two distinct flight schedules, impacted by two different kinds of weather-related disruptions (i.e. uncertainty from indeterminate aleatoric features), which represent two separate samples from *disrupted* test (unseen) data set, by employing the UTFM (*operation-schedule* framework) for airline disruption management. We selected these flight schedules as candidate test subjects for our demonstration because they represent major routes in the network of the US airline carrier that provided the data which enabled the development of the UTFM. For our assessments, we implement an aggregate non-ergodic HMM representation of the UTFM, such that the disruption management process strictly starts at Turnaround Schedule (TAS) phase of activity and ends at Taxi-In Outcome (TIO) phase of activity.

Figure 4.10 shows the probabilistic graphical model representation of the UTFM for disruption management on the operation of a typical flight from Dallas to Houston (DAL-HOU), which was disrupted by air traffic control (ATC) hold for bad weather at Dallas (i.e. *HDO6* delay code). Figure 4.10 reveals that there is a 100% likelihood that a specialist agent transitions to employ reactive disruption management measures from tactical disruption management measures during the turnaround phase of flight operation at Dallas (100% transition probability from TAS to TAD). As such, to effectively resolve the same disruption instance in the future, the most likely approach is adjust or update features in the turnaround, taxi-out and enroute phases of flight operation accordingly, as evidenced by internal state probabilities of 16%, 6%, and 3% for remaining in the TAO, TOO, and EO phases of activity respectively. Furthermore, Figure 4.10 reveals that tactical disruption management measure implemented for the turnaround flight phase to address the ATC hold for inclement weather at Dallas for that particular Dallas to Houston flight was ineffective, as evidenced by the lack

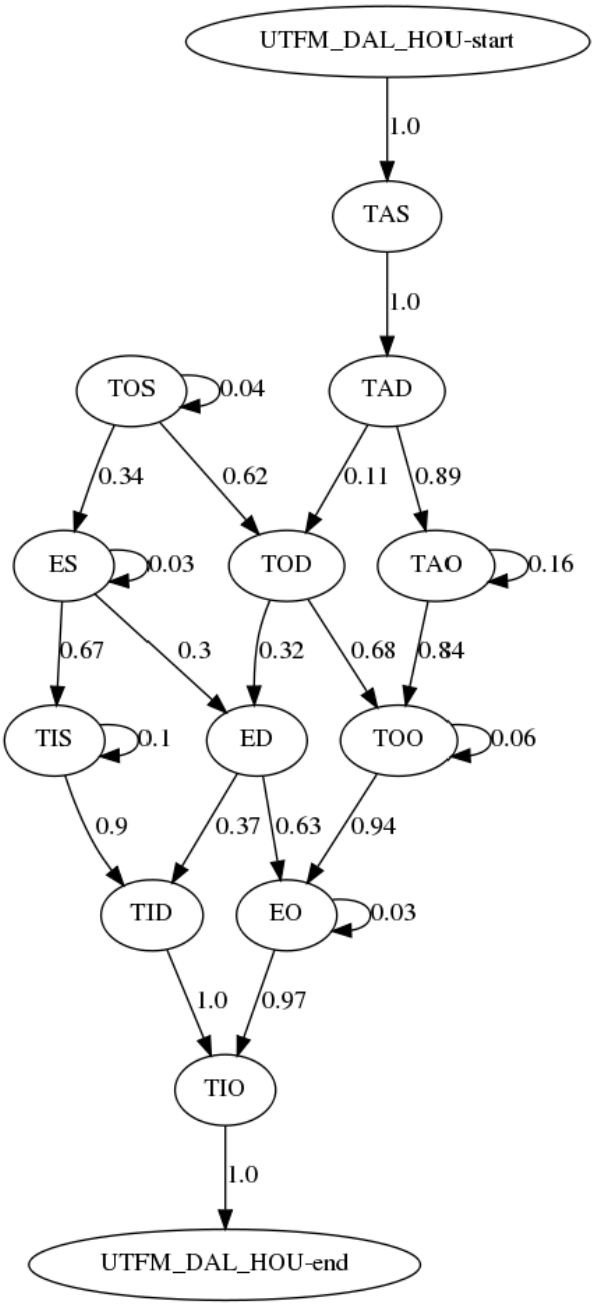

Figure 4.10. Probabilistic graphical map for UTFM assessment of a specific disrupted DAL-HOU flight

of transition from the turnaround phase of flight operation to the taxi-out phase of operation (i.e. zero probability of transition from TAS to TOS). As such delays were most likely incurred during the turnaround phase of operation while executing

that particular flight from Dallas to Houston. However, tactical initiatives proved somewhat effective during the taxi-out, enroute, and taxi-in phases of activity for disruption management of the Dallas to Houston flight, affirmed by internal state probabilities (i.e. interaction of hidden data features in *Intra-State* HMMs) of 4%, 3%, and 10% for remaining in the TOS, ES, and TIS phases of activity respectively.

Figure 4.11 shows the probabilistic graphical model representation of the UTFM for disruption management on the operation of a typical flight from Chicago to Boston (MDW-BOS), which was disrupted by ATC hold for bad weather en route to or at Boston (i.e. *HDO7* delay code). Figure 4.11 affirms that it is more likely that the tactical disruption management measures a specialist agent employs for disruption management of bad weather at Boston are proactively effective for the turnaround and taxi-out phases of flight operation, as indicated by internal state probabilities of 0.16 and 0.57 for TAS and TOS respectively and zero likelihood of transitions from those states to TAD and TOD respectively. Even though the tactical disruption management measures for addressing the inclement weather disruption at Boston in the enroute and taxi-in phases of activity are somewhat effective, there may be situations where decision-making for reactive disruption management at the enroute and taxi-in phases of activity during schedule execution may prove useful; as evidenced by the state transition probabilities of 0.16 and 0.59 from ES to ED and TIS to TID respectively. Furthermore, Figure 4.11 reveals that the proactive tactical disruption management measures for the turnaround and taxi-out phases of operation, implemented prior to departure from Chicago, were optimally effective for resolving ATC delay at Boston, as there are no transitions from TAS to TAD and TOS to TOD phases on activity in the UTFM. As such delays during the flight were accrued at the enroute and taxi-in phases of operation during disruption management. However, the UTFM representation from Figure 4.11 reveals that strategic disruption management initiatives to improve the future disruption resolution for this particular flight from Chicago to Boston, due to uncontrollable aleatoric uncertainty from inclement weather at Boston, do exist for turnaround, taxi-out and enroute phases

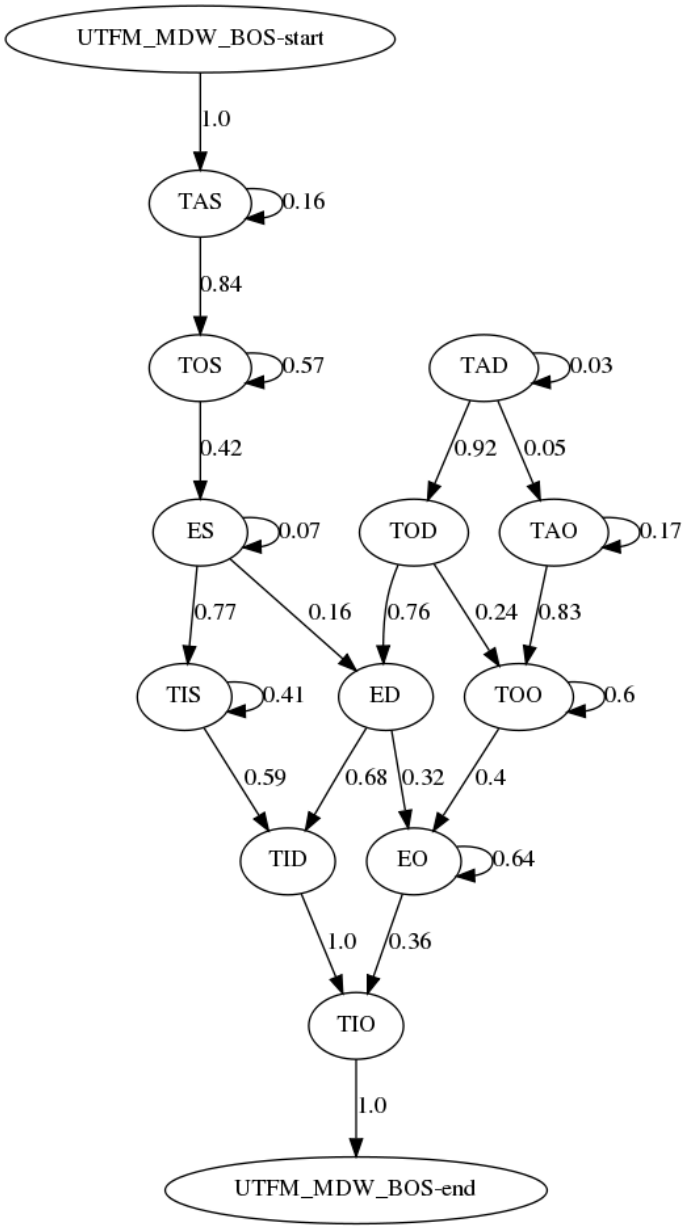

Figure 4.11. Probabilistic graphical map for UTFM assessment of a specific disrupted MDW-BOS flight

of flight operation; as indicated by internal state probabilities of 17%, 60%, and 64% for remaining in the TAO, TOO, and EO phases of activity respectively.

4.3 Predictive Transfer Function Model (PTFM)

Many models for the processes that affect different flight phases are defined by explicit objective functions, decision variables, and constraints, through optimization methods that impose a set of airline business rules at each phase of flight schedule execution from aircraft boarding at the origin airport to aircraft gate-parking at the destination airport [2]. As such, these optimization methods place an inherent restriction on the search rules that are used to obtain a feasible solution for a flight schedule feature, thereby enabling a parochial routine for estimating values of relevant flight schedule features during disruption management. Some of these models provide decent estimates of different flight schedule features for reactive disruption management during schedule execution [20, 114, 115]. However, their solution processes require considerable amount of time to evaluate the objectives and constraints posed by unscalable local and global routines that define the optimization of a monolithic system.

Current research and practices on airline disruption management have primarily focused on minimizing flight delay propagation in the air transportation network during reactive disruption management, by attempting to predict flight delay and its effect on flight schedule management by heuristically studying rudimentary mechanisms (i.e. rules) of operation in the air transportation network [116–119]. Moreover, the recent emergence of data-driven methods have provided avenues to readily discover and approbate air transportation mechanisms that enable the estimation of flight delay directly through data mining, in lieu of exploring existing flight delay propagation mechanisms [120]. As such, the status quo for evaluating the performance of disruption resolutions during airline disruption management rely on the precision and accuracy in estimating flight delay duration. While flight delay duration provides a credible performance metric (or proxy) for assessing reactive disruption management (i.e corrective actions) during schedule execution, it can not provide the criteria necessary to readily evaluate the performance of disruption management ini-

tiatives obtained and implemented during the strategic and tactical phases of disruption management prior to schedule execution. In that regard, many airlines (including Southwest Airlines) spend considerable amount of time and resources in scheduling block time and turnaround duration during proactive airline scheduling before schedule execution, in order to alleviate flight delays during schedule execution on day of operation [111].

For many years, senior management in the AOCC at many airlines have argued that the measure of the performance of a corrective action for resolving a specific disruption event is only viable for reactive disruption management during schedule execution. As such, there are several industry-wide metrics that human specialists in the AOCC use to examine the quality of their corrective actions for disrupted flight schedules during irregular operations. Some of these metrics include [47]:

- Re-accommodation time period required for all passengers on a disrupted flight schedule
- Time period required to create a plan of delays (or cancellations) that will enable the resumption of a disrupted flight schedule.
- The capacity for a disrupted flight schedule to depart exactly as scheduled, and arrive on schedule or within $14mins$ of original schedule.
- The amount of delayed or canceled flight legs in a flight schedule.

Although each of the aforementioned metrics is relevant for improving the quality of disruption management solutions over time, every one of them only addresses an aspect of mitigation approaches for irregular operations during schedule execution. As such, to enable an objective and all-inclusive methodology that conclusively measures the performance of irregular operations for a disrupted flight schedule, it is imperative that a candidate intelligent agent for airline disruption management is capable of expediently estimating the performance of alternative disruption resolutions (or flight schedules) by applying multiple performance metrics. To that effect,

supervised learning techniques provide a suitable medium for creating models that can promptly estimate useful target features for appraising the performance of disruption resolution initiatives at different phases of disruption management during the airline scheduling process. Thus, our proposed predictive transfer function model (or PTFM) offers a complete routine that mines historical data on flight schedules and their corresponding performance on airline resource management to calibrate functional structures (i.e., find and apply new search rules), which represent efficient tools for an intelligent domain manager. These tools enable prompt and precise estimation of appropriate measures of performance for disruption resolutions (i.e. alternate flight schedules) during strategic, operational, and tactical phases of disruption management.

4.3.1 PTFM Methodology

From a passenger's perspective, considerable evaluation of the quality of service provided by an airline during the journey from an origin airport station to a destination airport station starts from aircraft boarding at the departure station's gate and ends upon aircraft parking at the arrival station's gate [110]. As such, airlines aim to ensure optimal efficiency in the operational processes that ensue during the events in the flight execution horizon shown in Fig. 4.12.

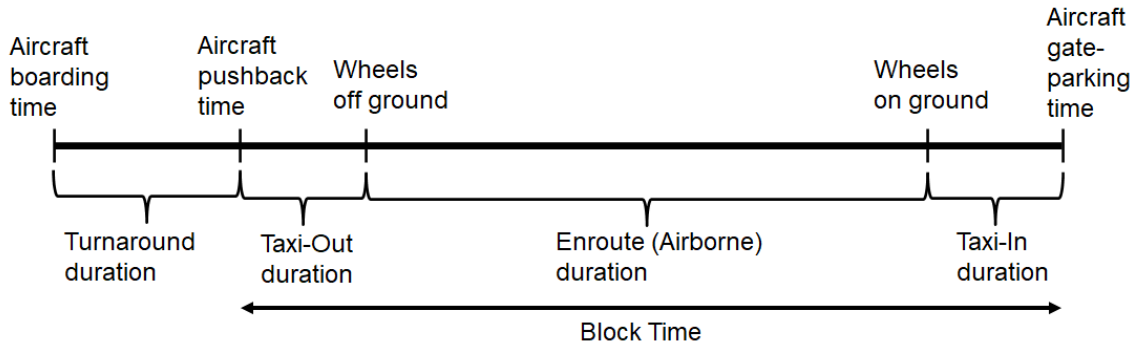

Figure 4.12. Timeline horizon during flight schedule execution

Hence, there are two primary stages during the execution of a characteristic flight schedule on day of operation, based upon an airline's resource capacity to serve its customers (i.e. passengers). The first stage, known as *flight capacity management*, defines the airline's ability to effectively load and unload an aircraft with passengers and necessary flight provisions (such as food and drinks), while intermittently conducting minor aircraft servicing tasks. Thus, from an airline operations perspective, the main objective of flight capacity management is to precisely estimate the time period required to complete the aircraft boarding (and deplaning) process during flight schedule execution, which is also known as turnaround duration as shown in Fig. 4.12. As such, a secondary objective of flight capacity management is to minimize unnecessary holdup (i.e. tactical delay) during the turnaround process.

The second stage of flight schedule execution also known as *flight service management* defines the capacity of the airline and other air transportation stakeholders (such as air traffic control) to efficiently move the aircraft (loaded with passengers) from a designated gate at the departure airport station to a paritcular gate at the arrival airport station. To this effect, flight service management can be divided into three separate periods during schedule execution as shown in Fig. 4.12, based upon the management of aircraft operations, namely: Taxi-out, Enroute, and Taxi-in [110]. Taxi-out represents the time and process used to move the aircraft from the gate to the runway at the departure airport prior to takeoff, enroute represents the duration and process where the aircraft is airborne as it makes its way to the destination airport, and taxi-in represents the duration and process used to transport the aircraft from the runway to the gate at the arrival airport. As such, the main objective of flight service management is to accurately estimate the duration (i.e block time) between the time when the aircraft pushes back from the departure gate at the origin airport and the time when the aircraft parks in the arrival gate at the destination airport. In complement, a secondary objective of flight service management is to minimize discretionary holdup (i.e. strategic delay) during the pushback of an aircraft from the departure airport gate to the parking of the aircraft at the arrival airport

gate during irregular operations. To this end, the estimation of turnaround duration and block time present a predictive modeling problem for airline resource capacity (i.e. flight capacity and flight service) management during schedule execution and disruption management.

Problem Formulation as an Artificial Neural Network

We formulate our PTFM framework for airline disruption management based upon the underlying principles of an artificial neural network (ANN). The artificial neural network is a data structure inspired by a framework of biological neurons in living organisms like the human brain, wherein each neuron is a unit that performs a simple task characterized by responding to an input signal. However, a connected network of neurons (such as the human brain) is capable of completing complex tasks and processes with impeccable speed and accuracy [121]. Similar to the biological neural network, an ANN for airline resource capacity management during irregular operations represents a connection of nodes that are analogous to neurons that implicitly describe the physical processes of airline disruption management. To this effect, the ANN is defined by three pertinent characteristics namely: node character, network topology, and learning rules [122].

- **Node character** represents the information (signal) processing qualities of a node, which are properties such as number of inputs and outputs associated with a node, the appropriate weight for each input and output, and the node's activation function. Fig. 4.13 shows a generic model for a single neuron or node in an ANN for estimating relevant flight schedule features for optimal airline schedule execution during disruption management. The node, which represents a specific flight schedule or disruption feature, obtains multiple inputs from other nodes or flight schedule features that define associated weights describing the strength (or importance) of the flight schedule features with respect to the node. To enable the transmission of information amongst flight schedule

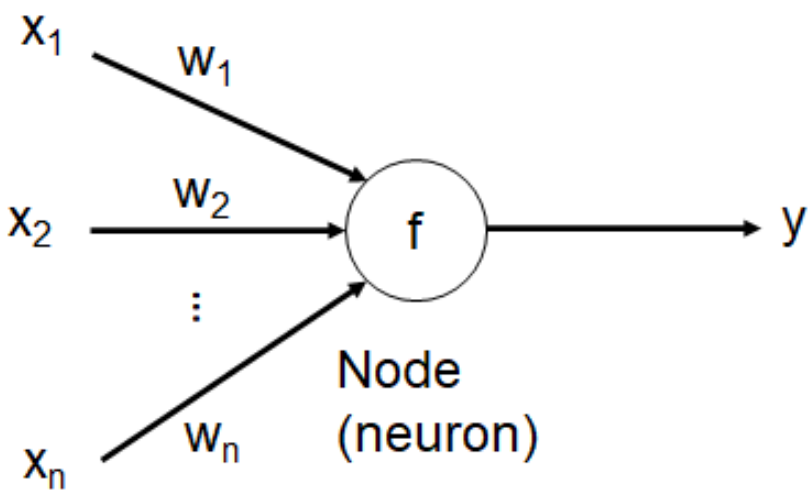

Figure 4.13. Fundamental model of a single neuron in an artificial neural network

features, a transfer function is used to determine the activation of the flight schedule feature (node) when the weighted sum of the inputs from other flight schedule features exceeds a certain threshold value, as shown by the expression in Eqn. 4.6

$$y = f(\sum_{i=0}^{n} w_i x_i - T) \tag{4.6}$$

where y is the response of the flight schedule feature (i.e information recipient node), f represents the transfer function, w_i is the weight or importance of an input flight schedule feature x_i, and T is an arbitrary threshold value. The expression for the transfer function in Eqn. 4.6 can be defined by linear and non-linear functions. However, non-linear functions such as the sigmoid function $S(x)$ [123], generally expressed as:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{4.7}$$

are often preferred for modeling real world applications because of their continuous differentiable property [124].

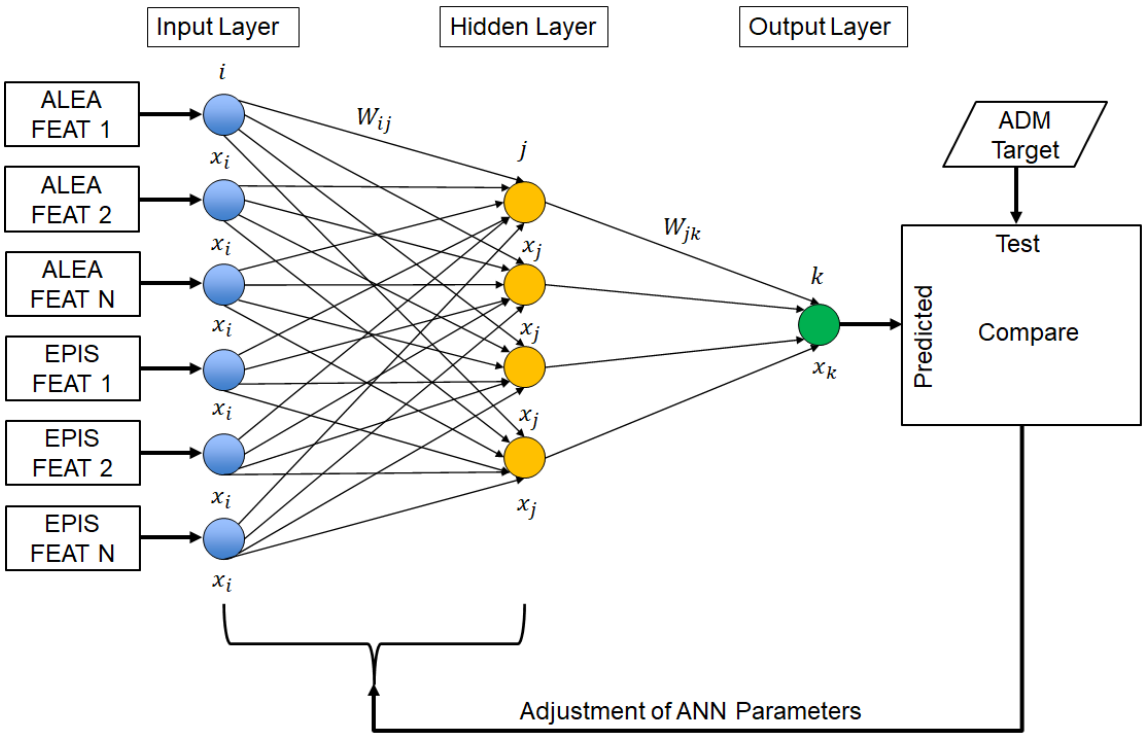

Figure 4.14. Artificial neural network framework for estimating target flight schedule features

- **Network topology** of an ANN defines the manner in which nodes representing flight schedule features are organized and connected in a neural network, thereby defining the general architecture of the ANN. A single hidden layer perceptron provides an excellent medium for solving linearly separable problems and can also be used to define a Gaussian process [125–127]. As such, we define the topology of our artificial neural network as a single hidden layer perceptron, based upon the following properties gleaned from the exploratory data analysis discussed in Chapter 3:

  1. Determinate aleatoric and epistemic schedule features in the data set are linearly separable as demonstrated by the existence of orthogonal linear combinations of flight schedule features through principal component analysis (PCA).

2. The functional process at any stage of airline resource capacity management (such as turnaround) can be defined by a Gaussian process, as demonstrated by the existence of optimal kernel functions and hyperparameters from Gaussian process regression (GPR).

Fig. 4.14 shows the ANN framework for estimating appropriate target features like turnaround duration for disruption management. The ANN framework, shown in Fig. 4.14, represents a feedforward perceptron that consists of an input layer i with a set of aleatoric and epistemic flight schedule features, a hidden layer j that transmutes information from nodes in the input layer, and an output layer k with a flight schedule feature that represents performance measures at any stage of flight schedule execution. For validation of the ANN topology, predicted values of performance measures are compared with actual (realized) values to inform the adjustment of the ANN parameters for improving its predictions.

- **Learning rules** define the routines that calibrate an ANN by optimizing the parameters of the network topology (or structure) through the use of data that represents instances of flight schedule executions. For calibration, we adopt a supervised learning approach wherein the ANN is trained first before it is applied only when the optimized network topology produces the desired performance from a target feature based upon a set of input features. As such, supervised learning ensures that all possible search rules for obtaining reasonable estimates of target features are inherently cached and readily accessible during irregular operations, thereby reducing the time required for disruption management. Fig. 4.15 reveals a generic process for learning a suitable ANN to facilitate the prediction of pertinent features for airline resource management during irregular operations. The process begins by defining input and output flight schedule features appropriate for each stage of airline resource management, and then dividing available instances of flight schedule executions into

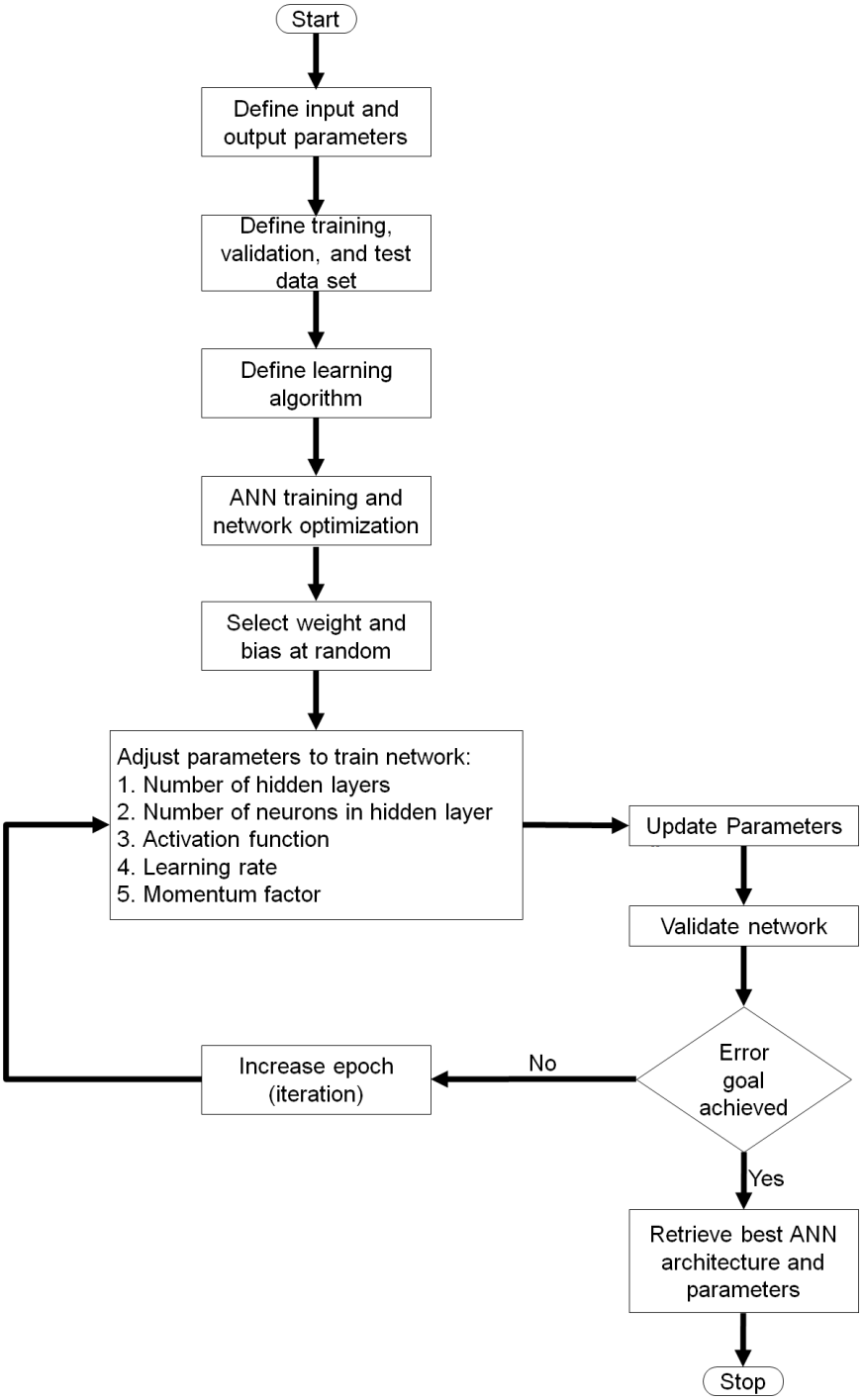

Figure 4.15. Flowchart of general process for ANN learning

training and testing categories. Next, a learning algorithm is defined and the optimal parameters for the network topology of the ANN are estimated based upon error correction methods that employ a backpropagation mechanism [128]. An error function can be defined as the difference between the flight schedule feature (node) value in the output layer and a corresponding target flight schedule feature value. Let $y_{k,n}$ be the value of the data feature (i.e. node) in the k_{th} output layer at epoch n during training, and y_k^* be the target value for the data feature in the k_{th} output layer. Thus, the error function is defined as:

$$e_k = y_{k,n} - y_k^* \tag{4.8}$$

To define an error goal, let θ be a constant positive value (i.e. learning parameter) that regulates the rate at which the weights of the network are adjusted based upon the following expression:

$$w_{kj,n+1} = w_{kj,n} - \theta e_k x_j \tag{4.9}$$

Eqn. 4.9 describes the new weight vector at the next epoch, $w_{kj,n+1}$, for an input data feature x_j that reduces the error value at epoch $n+1$ based upon error function e_k. As such, θ defines the rate at which the ANN learning process shown in Fig. 4.15 converges.

4.3.2 Solution Approach for PTFM

Similar to the UTFM, we employ a component assembly process to create artificial neural networks for predicting different target features for flight capacity management and flight service management at all phases of airline disruption management. By applying generative programming techniques, specialized ANNs for each functional role in the AOCC are trained and validated through supervised learning, before performance measures that represent the outputs of the PTFM are estimated by combining the prediction of specific flight schedule features by employing trained component ANNs. Fig. 4.16 shows our solution approach for automatic estimation of perfor-

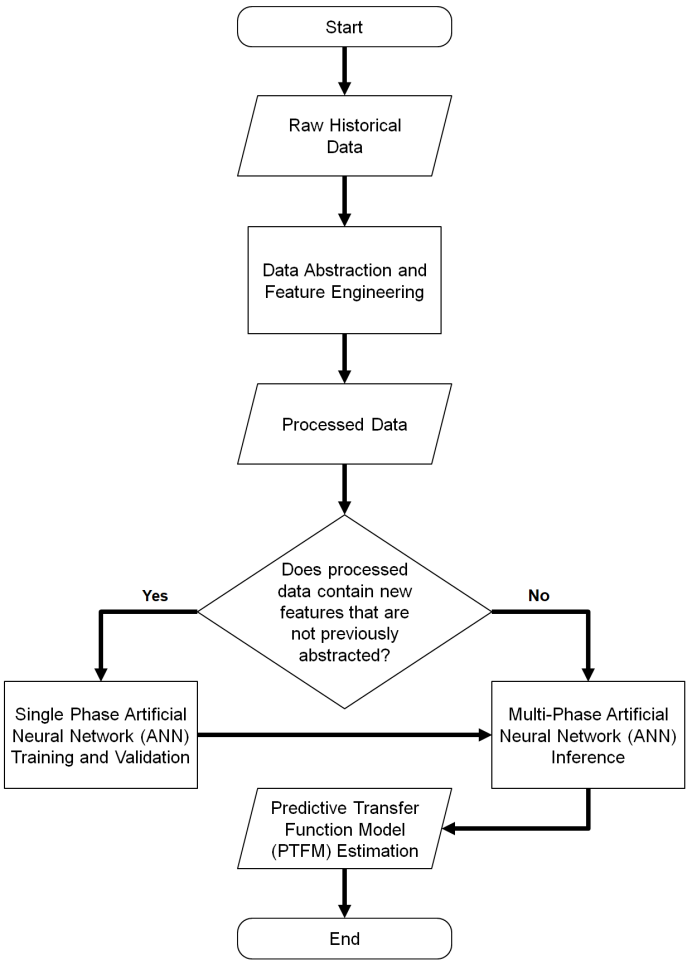

Figure 4.16. Component assembly approach for automatic estimation of performance measures for disruption management

mance measures for airline disruption management. Just like the UTFM solution process, the approach begins by preprocessing raw data into readily decipherable data formats that are suitable for ANN learning algorithms, by employing the techniques described in Section 4.1. Next, we learn the optimal network topology of each ANN for predicting target flight schedule features at each stage of airline resource capacity management, before performance measures at each stage are estimated through a combination of flight schedule features inferred from the optimal ANNs.

For the remainder of this section, we discuss the methods employed to define and learn optimal structures of separate ANNs that represent components of the

PTFM for airline disruption management. Next, we demonstrate the inference of performance measures from the PTFM based upon a simple set of overarching rules for disruption management.

4.3.3 Single Phase ANN Development

Single phase ANN development represents the creation of artificial neural networks for each of the three individual phases of airline disruption management namely: proactive disruption management before schedule execution (i.e. tactical disruption management), reactive disruption management during schedule execution (i.e. operational disruption management), and proactive disruption management after schedule execution (i.e. strategic disruption management). By applying guidelines defined in Fig. 4.15 from Section 4.3.1, we discuss key routines and rationale for developing customizable ANNs for each phase of airline disruption management.

Defining Inputs and Targets

For each of the three phases of disruption management, there exist relevant flight schedule features in a historical airline scheduling and operations data set that are of significant interest to AOCC practitioners. With respect to the processes involved during flight schedule execution, tactical disruption management aims to eradicate any inconvenience to passengers that may arise during flight capacity management of irregular operations prior to aircraft pushback from the departure airport gate. As such, turnaround duration represents an appropriate target flight schedule feature for a feedforward ANN for tactical disruption management. Operational disruption management seeks to ensure that the departure and arrival flight chronology retrospectively set by airlines (and sanctioned by other air transportation stakeholders) remains valid during schedule execution. To that effect, binary indicators for on-time flight arrival and arrival within $14mins$ of original schedule (i.e. $A0$ and $A14$ respectively) represent suitable target flight schedule features for a feedforward ANN

for operational disruption management. Complementary to tactical disruption management, strategic disruption management aims to eradicate operational delays and inconsistencies during flight service management of irregular operations after aircraft pushback from the departure airport gate. Thus, block time represents a suitable target flight schedule feature for an ANN for strategic disruption management. From a feature abstraction perspective, the target flight schedule features for all three phases of airline disruption management represent epistemic flight schedule features. Inputs for the artificial neural networks for each of the three phases of airline disruption management are defined based upon appropriate aleatoric and epistemic features that are observable at different stages of flight schedule execution. As such, all determinate aleatoric features (i.e. features representing flight date, origin airport, destination airport, number of passengers, etc.) are viable ANN inputs that indicate the uniqueness of a particular flight schedule with respect to the airline route network. Furthermore, indeterminate aleatoric features (such as features representing IATA delay codes) are necessary ANN inputs for estimating on-time performance of reactive disruption management during schedule execution, and define the uniqueness of a particular flight schedule with respect to the disruptions encountered in the airline route network. Lastly, epistemic features provide inputs that indicate the uniqueness of the disruption resolution (or *rule-of-thumb*) applied by a human specialist in the AOCC during irregular operations and disruption management.

Data Segmentation for Learning

Prior to invoking suitable algorithms for learning the optimal parameters of the network topology of ANNs for each phase of disruption management, it is necessary to define the data set that will inform the learning and validation process based upon the general objectives for each phase. Recall from Chapter 3 that there are two distinct chunks of data that make up the full data set namely: *non-disrupted* and *disrupted* data sets. The *non-disrupted* data set represents all instances of flight schedules in

the Southwest network that executed without any disruptions between September 2016 and September 2017, and the *disrupted* data set represents all instances of flight schedules that executed with disruptions (i.e. delays) during the same time frame. As such, two separate ANNs that define the estimation of actual turnaround duration for proactive disruption management before schedule execution are created by using the *non-disrupted* data set and *disrupted* data set respectively. Similarly, two separate feedforward ANNs that define the estimation of actual block time duration for proactive disruption management after schedule execution are developed from the *non-disrupted* and *disrupted* data sets respectively. For evaluation of on-time performance for reactive disruption management during schedule execution, two separate ANNs that predict on-time aircraft arrival and aircraft arrival within $14mins$ at the destination airport gate, respectively, are created using the *disrupted* data set. Thus, a total of six distinct feedforward ANNs are created across all three phases of airline disruption management. The appropriate data set for developing each of the ANNs is further partitioned, via a random seed of 42, such that 70% of the instances of flight schedule execution in the data set are randomly selected to train the ANN, and the remaining 30% of flight schedule executions are used to test the validity the trained ANN.

ANN Learning and Validation

Upon defining the inputs, targets and appropriate data sets for learning the ANNs for each phase of disruption management, the next step in the single phase ANN development process is to find optimal parameters for the network topology that defines the ANNs. Thus, we define the network topology for each of the six ANNs as the single hidden layer perceptron shown in Fig. 4.14. All nodes in input and output layers of the ANNs are activated by linear functions, while nodes in the hidden layer are activated by nonlinear functions described later on in this section. The optimal size (i.e. number of nodes) of the hidden layer is usually between the size of the input

and size of the output layers [129]. Hence, we set the number of nodes in the hidden layer of the perceptron as the mean of the number of nodes in the input layer and the number of nodes in the output layer for each of the six ANNs.

1. *Learning*: ANNs for tactical and strategic disruption management have continuous targets (i.e. turnaround duration and block time respectively). As such, learning optimal weights for these ANNs presents a regression problem. Thus, we define the error function for adjusting the weights of the single hidden layer perceptron as the Huber loss function (H) [130], expressed as follows:

$$H(x, y) = \frac{1}{n} \sum_{i}^{n} z_i \tag{4.10}$$

such that z_i is given by:

$$z_i = \begin{cases} 0.5(x_i^2 + y_i^2) & \text{if} \quad |x_i - y_i| < 1, \\ |x_i - y_i| - 0.5 & \text{otherwise} \end{cases} \tag{4.11}$$

where x_i and y_i represent each entry in the input space and target space of the training data set respectively, and n represents the total number of instances of flight schedule executions in the training data set. As shown in Eqns. 4.10 and 4.11, the Huber loss function creates an error goal that uses a squared term if the absolute element-wise error is below 1 and an $L1$ term otherwise, thus making it amenable (i.e. less sensitive) to accommodating for outliers during ANN training. We employ the logarithm of the sigmoid function to activate the nodes in hidden layers of the ANNs for tactical and strategic disruption management during the search for optimal network parameters. ANNs for operational disruption management have binary targets (i.e. 0 or 1) for whether or not an aircraft arrives at the destination airport gate at a particular time. As such, learning optimal weights for these ANNs presents a classification problem. To that effect, we define the error function for adjusting the weights of the single hidden layer network topology for operational disruption management as

a function that measures the binary cross entropy (BCE) [131] between the target and output layer value, expressed as follows:

$$BCE(x, y) = \frac{1}{n} \sum_{k}^{n} b_k \tag{4.12}$$

such that b_k is given by:

$$b_k = -[y_k \log x_k + (1 - y_k) \log (1 - x_k)] \tag{4.13}$$

where x_k is the value of the node in the output layer of the ANN and y_k is the target value adjusted by the sigmoid function. The nodes in the hidden layer of ANNs for operational disruption management are activated through the softplus function $s(x)$, expressed in Eqn. 4.14, to obtain the weights that optimally define the network topology during training.

$$s(x) = \log (1 + e^x) \tag{4.14}$$

We apply the popular adaptive moment estimation (Adam) learning algorithm [132] to find the optimal parameters of each of the six ANNs, by ensuring that the error goal for training the respective ANNs remains constant after a considerable amount of epochs.

2. *Validation*: After learning the optimal weight parameters of the network topology for each of the six ANNs for separate phases of disruption management, it is imperative to verify the credibility of the estimations from the neural networks. The ANNs that address the regression problems for tactical and strategic disruption management are validated by simply comparing the actual turnaround and block time duration values from appropriate test (unseen) data sets with the turnaround and block time duration values predicted by the ANNs, based upon corresponding input flight schedule features from the test data sets. As such, the root mean square error (RMSE) indicates the absolute fit of a generic ANN's predicted values with observed test data values, by providing a measure

of the standard deviation of the unexplained (residual) variance in the predictive capacity of the ANN [133]. RMSE also has the useful property of being in the same units as the target values from the test data and is expressed as follows:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i^2 - y_i^2)}{n}} \tag{4.15}$$

where y_i represents a target value in a test data set, $\hat{y}_i$ indicates the corresponding predicted value from the ANN, and n represents the total number of samples in the test data set.

For ANNs that solve the classification problem for operational disruption management, a special parameter called area under the receiver operating characteristic (ROC) curve (or AUC for short) is used to test the validity of the ANNs. The ROC curve is a two-dimensional graph that reveals the performance of a classification model for any classification threshold. This curve plots two parameters namely: true positive rate and false positive rate. True positive rate, also known as recall or sensitivity, is a function of correct ANN predictions of on-time arrival (i.e. $A0$ or $A14$) from the test data set that are identified (i.e true positive or TP) and incorrect ANN predictions of $A0$ or $A14$ from the test data set that are rejected (i.e. false negative or FN). False positive rate, also known as fallout, is a function of incorrect ANN predictions of on-time arrival that are identified from the test data set (i.e. false positive or FP) and correct ANN predictions of on-time arrival from the test data set that are rejected (i.e. true negative or TN). As such, recall and fallout are expressed as follows:

$$recall = \frac{TP}{TP + FN} \tag{4.16}$$

$$fallout = \frac{FP}{FP + TN} \tag{4.17}$$

Thus, AUC measures the entire two-dimensional area underneath an ROC curve from (0,0) to (1,1). AUC values range from 0 to 1, such that an ANN whose predictions are 100% wrong has an AUC of 0.0 while an ANN whose predictions are 100% correct has an AUC of 1.0. Some other metrics that can be used to verify the credibility of the ANNs for operational disruption management are precision and F1-score [134], which assess correctly identified estimations of on-time arrivals and the harmonic mean of precision and sensitivity of on-time arrivals respectively.

4.3.4 Multi-Phase ANN Inference

Multi-phase ANN inference represents the appropriate estimation of turnaround duration and block time for both stages of airline resource management during irregular operations and disruption management. As mentioned in Section 4.3.3, the estimation of separate features of interest at each phase of airline disruption management relies on multiple ANNs. As such, we apply a parallel ensemble approach [135] to obtain quality estimations of turnaround duration and block time, such that the component ANNs for each phase of airline disruption management are independently learned in parallel and their corresponding predictions are combined via a custom bootstrap aggregating (i.e. bagging) procedure. The bootstrap aggregating approach leverages the independence between the component ANNs (i.e. base learners) to ensure that the overall error from the PTFM estimates of turnaround duration and block time can be significantly reduced by combining independent ANN predictions [135].

Fig. 4.17 reveals the parallel ensemble approach for multi-phase ANN inference to obtain PTFM estimates of turnaround duration and block time for airline resource management and flight schedule execution during irregular operations. The bagging process begins by obtaining necessary refined data features for a disrupted flight schedule, which represent inputs of separate ANNs for tactical, operational, and strategic disruption management. Next, the predictions of binary indicators for

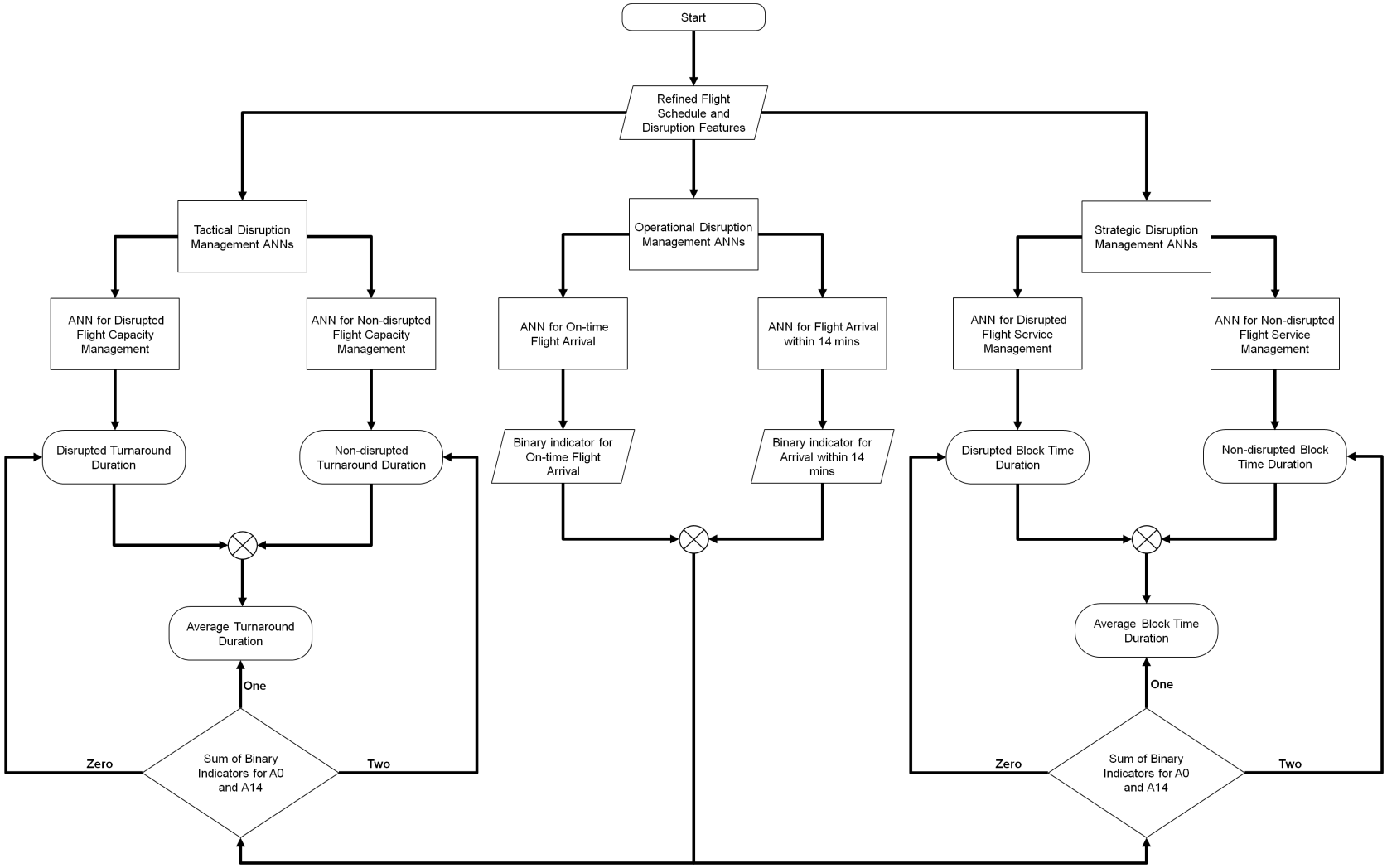

Figure 4.17. Flowchart of parallel ensemble approach for PTFM estimation

on-time flight (aircraft) arrival and flight arrival within $14mins$ of scheduled arrival, respectively, at the destination gate are retrieved through inference of the respective ANNs for operational disruption management. The inferred values of the binary indicators for both measures of on-time performance (i.e. $A0$ and $A14$) are subsequently summed, and used to retroactively estimate turnaround duration and block time through the inference of respective ANNs for tactical and strategic disruption management. A sum of zero (for $A0$ and $A14$) implies that a disrupted flight does not arrive as originally scheduled nor does it arrive within $14mins$ of the scheduled time, and the aircraft operated on the flight effectively arrives at the destination airport gate much later than $14mins$ past its original scheduled arrival time. A sum of $A0$ and $A14$ that yields one implies that the aircraft on a disrupted flight either arrives precisely as originally scheduled or within $14mins$ beyond the original scheduled arrival time at the destination airport gate. A sum of $A0$ and $A14$ that yields two indicates that a disrupted flight either arrives exactly as originally scheduled or within $14mins$ earlier than the precise arrival time scheduled for parking at the destination airport gate.

If the sum of the values for $A0$ and $A14$ is zero, then the turnaround duration and block time predictions from the tactical and strategic disruption management ANNs, trained by using the *disrupted* data set, are set as the PTFM estimations of turnaround duration and block time. If the sum of the values for $A0$ and $A14$ is one, then the averages of the turnaround duration and block time predictions from tactical and strategic disruption management ANNs that are trained by using the *disrupted* and *non-disrupted* data sets respectively, are set as the PTFM estimates of turnaround duration and block time. If the sum of the predicted values for $A0$ and $A14$ is two, then the turnaround duration and block time predictions from the tactical and strategic disruption management ANNs, trained by using the *non-disrupted* data set, are set as the PTFM estimations of turnaround duration and block time. Thus, we estimate the tactical delay incurred during flight capacity management as the difference between the turnaround duration predictions retrieved from the ANNs

for tactical disruption management, trained by using the *disrupted* and *non-disrupted* data sets respectively. Congruently, the strategic delay period accrued during flight service management is estimated as the difference between the block time predictions retrieved from respective ANNs for strategic disruption management, calibrated through the *disrupted* and *non-disrupted* data sets

4.3.5 Computational Setup

We now discuss the computational setup and analysis for the predictive transfer function model in a characteristic computational and intelligent agent (i.e functional role) in the AOCC. Table 4.4 summarizes specific epistemic input flight schedule features and corresponding target features in all ANNs for separate phases of airline disruption management.

These features represent inputs and targets that are applicable for separate classes of ANNs for each phase of disruption management, and are adopted based upon the results observed from exploratory data analysis. Furthermore, all determinate aleatoric flight schedule features represent additional generic inputs for all ANNs for

Table 4.4. Epistemic features for multi-phase ANN inference in PTFM

ADM Phase	**Epistemic input Features**	**Target Feature**
Tactical	*ADJST_TURN_MINS*	*ACTL_TURN_MINS*
Operational	*shiftper_sched_PB, shiftper_sched_GP, DOT_DELAY_MINS*	*A0, A14*
Strategic	*shiftper_actl_PB, shiftper_actl_GP, actl_enroute_mins*	*actl_block_mins*

each phase of disruption management. In complement, indeterminate aleatoric features (i.e. disruption features) represent additional specific inputs for ANNs that define the PTFMs for different functional roles in the AOCC during operational disruption management. Complete lists of the definitions of all aleatoric and epistemic features used for PTFM development can be found in the Appendix A. Following appropriate data preprocessing and segmentation, all ANNs are subsequently implemented through parallel learning (for 15,000 epochs) and then inference in the Python programming language, which is significantly accelerated through computations via the *PyTorch* software [136] running on an NVIDIA GTX 1080Ti graphics card [137].

4.3.6 PTFM Analysis and Results

This section is comprised of two separate parts, which describe different analyses and results from the development and implementation of the PTFM respectively. The first part on PTFM development outcome discusses the evaluation of the optimal ANN topology, obtained through learning, for the PTFM of a generic intelligent agent at each phase of disruption management. The second part on PTFM implementation results demonstrates the assessment of PTFM estimations from a generic intelligent agent obtained through the combination of the predictions from multiple optimal ANNs.

- **Single Phase Results**

  Fig. 4.18 shows the plots of the predictions of actual block time duration versus the observed block time duration for instances of non-disrupted flight schedules and delayed flight schedules due to weather disruptions respectively. The plot on the left in Fig. 4.18 represents the predicted block time duration from the ANN for strategic disruption management, learned by using the *non-disrupted* data set of instances of flight schedule executions with significantly more examples (430,000 training samples and 186,000 test samples). Thus, the left plot in Fig. 4.18 reveals that the predictive capacity of the ANN for non-disrupted flight

service management during strategic disruption management has a standard deviation of unexplained variance (i.e. RMSE) of $4.23min$ between the pushback of a particular aircraft at an origin airport and parking the aircraft at the destination airport gate. The plot on the right in Fig. 4.18 shows the predicted block time duration from the ANN for strategic disruption management, learned by using the *disrupted* data set with about 6,200 instances of delayed flight schedules due to inclement weather events. Unlike the left plot in Fig. 4.18, the plot on the right reveals a relatively lower predictive capacity for disrupted flight service management during strategic disruption management of weather-related delays, with a root mean square error of about $13mins$ between aircraft pushback and aircraft gate-parking for a specific flight.

Fig. 4.19 represents the plots of the predictions of actual turnaround duration versus the observed turnaround duration for instances of non-disrupted flight schedules and delayed flight schedules due to weather disruptions respectively. Similar to Fig. 4.18, the plot on the left in Fig. 4.19 represents the predicted turnaround duration from the ANN for tactical disruption management, learned by using the *non-disrupted* data set of instances of flight schedule executions. Hence, the left plot in Fig. 4.19 reveals that the predictive capacity of the ANN for non-disrupted flight capacity management during tactical disruption

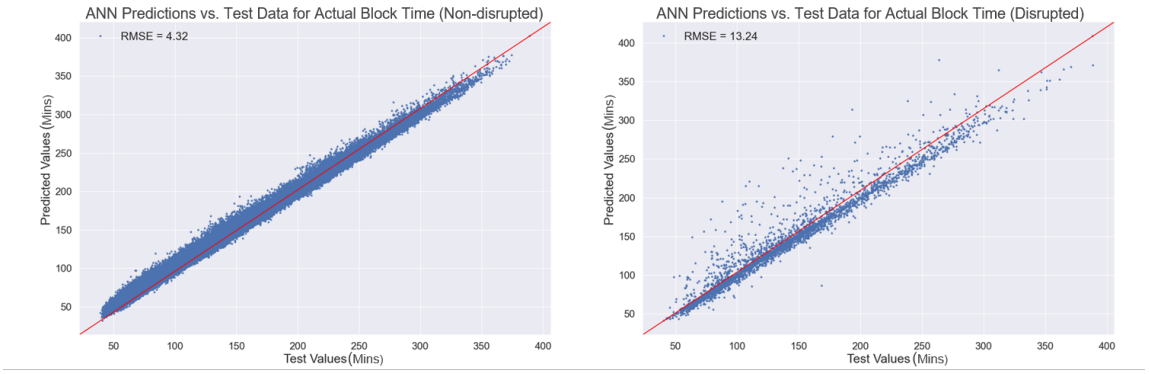

Figure 4.18. ANN predictions vs. test data for strategic disruption management

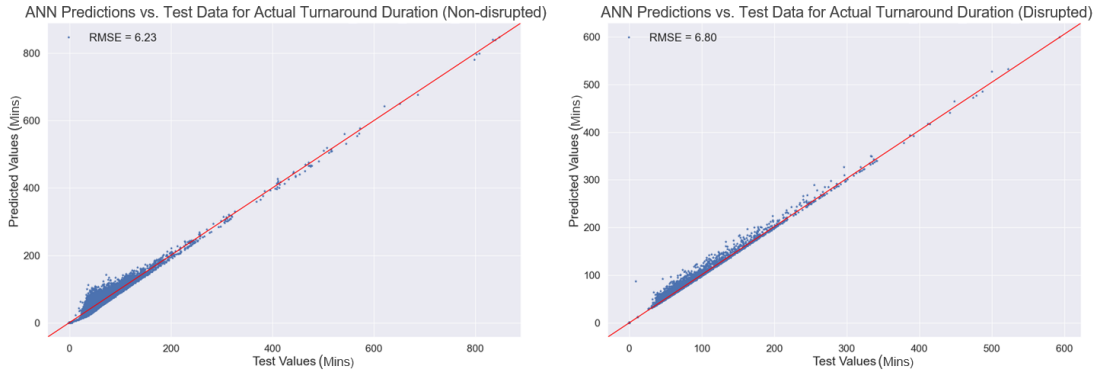

Figure 4.19. ANN predictions vs. test data for tactical disruption management

management has a standard deviation of unexplained variance (i.e. RMSE) of $6.23mins$ between the passenger boarding of a particular aircraft at an origin airport gate and pushback of the aircraft from the origin airport gate. The plot on the right in Fig. 4.19 shows the predicted turnaround duration from the ANN for tactical disruption management, learned by using the *disrupted* data set of instances of delayed flight schedules due to inclement weather events. Unlike Fig. 4.18, the plot on the right in Fig. 4.19 reveals a relatively similar predictive capacity for disrupted flight capacity management during tactical disruption management of weather-related delays, with a root mean square error of $6.80mins$ between aircraft boarding and aircraft pushback for a specific flight.

The red diagonal line in the plots in Fig. 4.18 and Fig. 4.19 represents the line of perfect prediction, such that coordinates (i.e. data points) that lie on this line indicate a perfect prediction of the observed test data value by the ANN. As such, Figs. 4.19 reveal near-perfect predictions of turnaround duration that are more than a couple of standard deviations away from the mean turnaround duration for both non-disrupted and disrupted flight schedules observed, respectively, in the test data sets.

Table 4.5. Model performance summary of ANNs for all functional roles in AOCC

Functional Domain	**Training Data Samples**	**Test Data Samples**	**Block Time RMSE (mins)**	**Turnaround RMSE (mins)**	**A0 AUC**	**A14 AUC**
Customer Hold	32,809	14,061	5.97	7.15	1.0000	1.0000
Dispatch CSC	12,228	5,240	8.79	6.58	1.0000	0.9996
Flight Operations	25,459	10,911	7.04	6.93	1.0000	1.0000
Fuel Management	3,389	1,452	8.02	5.88	0.9946	0.9929
Ground Operations	117,863	50,512	6.58	5.87	1.0000	1.0000
Inflight	55,611	23,833	6.99	5.63	1.0000	0.9999
Maintenance	23,463	10,055	7.21	6.70	1.0000	1.0000
NAS	15,851	6,793	8.97	6.85	0.9998	0.9999
Security	2,069	886	7.34	6.59	0.9849	0.9895
Technology	6,267	2,686	7.47	6.12	1.0000	1.0000
Weather	8,861	3,798	13.24	6.80	0.9972	0.9987

Fig. 4.20 shows the ROC plots of the separate ANNs for operational disruption management, which predict whether or not a specific disrupted flight arrives exactly on-time and within $14mins$ of the scheduled arrival time, respectively. Thus, both ANNs, described by Fig 4.20, are learned by using the *disrupted* data set of instances of delayed flight schedule executions due to weather-related disruptions. The blue dashed diagonal line in both plots in Fig. 4.20 represents perfect chance. That is, an ANN for operational disruption management that makes predictions based upon the blue diagonal line has no better odds of detecting $A0$ and $A14$ than a random coin flip. As such, the solid orange curve from the plot on the left in Fig. 4.20 represents the receiver operating characteristic (ROC) curve of the ANN classifier for assessing on-time arrival (i.e. $A0$) of a disrupted flight schedule due to weather-related disruptions. The area under the ROC curve (i.e. AUC) is 0.9972, which indicates that the ANN can correctly classify the on-time arrival status of a disrupted flight schedule due to weather-related delays over 99% of the time. Similar to the left plot, the solid orange curve from the plot on the right in Fig. 4.20 reveals the ROC curve of the ANN classifier for assessing arrival within $14mins$ (i.e. A14) of a disrupted flight schedule due to weather-related disruptions. As such, the area under the orange ROC curve, from the plot on the right in Fig. 4.20, is 0.9987,

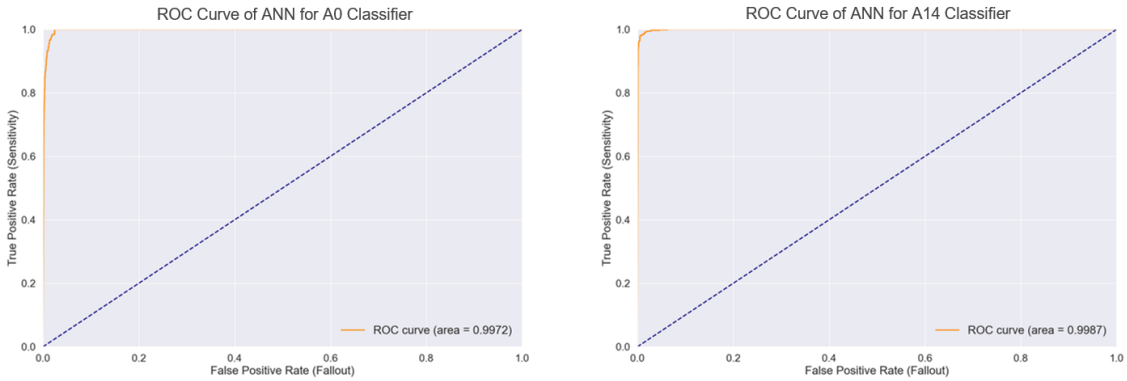

Figure 4.20. ROC plots of ANN classifiers for operational disruption management

thus indicating that the ANN can correctly identify whether or not a disrupted flight schedule arrives within $14mins$ of the original schedule over 99% of the time.

Table 4.5 shows the summary of the model performance of separate ANNs in estimating appropriate flight schedule features for all functional roles in the AOCC during disruption management. As shown in Table 4.5, all ANNs for operational disruption management have near perfect performance with AUC values of approximately 1. The highest RMSE value (i.e. $7.15mins$) for predicting turnaround duration for disrupted flight capacity management during schedule execution (i.e. tactical disruption management) was attained by the ANN for the Customer Hold functional role in the AOCC. Thus, this implies that the predictive capacity for disrupted flight capacity management during tactical disruption management of delays related to holding aircraft for passengers on inbound flight connections has a standard deviation of unexplained variance of $7.15mins$ between aircraft boarding and aircraft pushback for a specific flight. As evidenced from Table 4.5, the highest RMSE value for predicting block time for disrupted flight service management during schedule execution (i.e. strategic disruption management) is less than $14mins$, and attained by the ANN for the Weather functional role in the AOCC. Thus, the predictions from the ANNs for flight service management are valid with respect to current industry standards for on-time arrival.

- **Multi-Phase Results**

  We now employ the multi-phase inference of the set of ANNs for airline disruption management to demonstrate the PTFM estimation of turnaround duration and block time for a specific test flight schedule from Dallas to Houston, affected by delay due to air traffic control hold for bad weather in Dallas.

  Table 4.6 reveals the inputs of respective ANNs for different phases of airline disruption and resource management for the disrupted (test) flight schedule from

Dallas to Houston, same as the test case for UTFM assessment in Section 4.2.7. Features in Table 4.6 that serve as general inputs of all ANNs across separate phases of disruption management are determinate aleatoric features, which define the specificity (or uniqueness) of the test flight schedule being considered for our demonstration with respect to other flight schedules in the test data

Table 4.6. Input features and corresponding values for a specific disrupted DAL-HOU flight.

Input Feature	**ADM Phase**	**Standardized Value**
ADJST_TURN_MINS	Tactical	-0.63548
shiftper_sched_PB	Operational	-0.07008
shiftper_sched_GP	Operational	0.66334
DOT_DELAY_MINS	Operational	0.08585
shiftper_actl_PB	Strategic	0.39709
shiftper_actl_GP	Strategic	1.38748
actl_enroute_mins	Strategic	-0.49476
ATC Hold at Origin	Operational	1.12319
doy	All	-0.01179
orig_x_dir	All	-0.25736
orig_y_dir	All	-0.75922
orig_z_dir	All	-0.50746
dest_x_dir	All	-0.12307
dest_y_dir	All	-1.46375
dest_z_dir	All	-1.25591
ONBD_CT	All	0.46063
route_dist	All	-1.16903
sched_route_originator_flag	All	-0.36540

set. Features in Table 4.6 that serve as specific (additional) inputs of ANNs for each phase of disruption management are epistemic features, which define the observability and peculiarity of the disruption resolution of a particular flight schedule during schedule execution. As such, the values of epistemic features

Table 4.7. Target features and corresponding values for a specific disrupted DAL-HOU flight.

Target Feature	**Value Source**	**Actual Value**
Actual A0	Real Schedule Execution	0
Actual A14	Real Schedule Execution	0
Scheduled Turnaround (mins)	Real Schedule Execution	35.00
Actual Turnaround (mins)	Real Schedule Execution	**38.00**
Scheduled Block Time (mins)	Real Schedule Execution	65.00
Actual Block Time (mins)	Real Schedule Execution	**104.00**
Actual Tactical Delay (mins)	Real Schedule Execution	**3.00**
Actual Strategic Delay (mins)	Real Schedule Execution	**39.00**
Predicted A0	Operational ANN	0
Predicted A14	Operational ANN	0
Non-disrupted Turnaround (mins)	Tactical ANN	30.03
Disrupted Turnaround (mins)	Tactical ANN	38.37
Non-disrupted Block Time (mins)	Strategic ANN	92.02
Disrupted Block Time (mins)	Strategic ANN	93.61
Estimated Turnaround (mins)	PTFM Inference	**38.37**
Estimated Block Time (mins)	PTFM Inference	**93.61**
Estimated Tactical Delay (mins)	PTFM Inference	**8.34**
Estimated Strategic Delay (mins)	PTFM Inference	**1.59**

represent the inputs (or *rules-of-thumb*) applied by human specialists in the AOCC during disruption management.

Table 4.7 shows the predicted values of target features from separate ANNs for disruption management and actual values from the observed (i.e. real world) execution of the specific disrupted flight from Dallas to Houston, defined by the ANN input information from Table 4.6. As evidenced in Table 4.7, the ANNs for operational disruption management predicted that the flight would not arrive at Houston as scheduled nor within $14mins$ of the scheduled arrival time (i.e. binary indicators of 0), which is consistent with the observed (real) execution of that particular flight from Dallas to Houston. The ANNs for tactical disruption management predicted a turnaround duration of $30.03mins$ and $38.37mins$, repectively, for *non-disrupted* and *disrupted* flight capacity management of aircraft boarding at Dallas. In complement, the respective ANNs for strategic disruption management predicted a block time duration of $92.02mins$ and $93.61mins$ for *non-disrupted* and *disrupted* flight service management of aircraft operation from pushback at Dallas to gate-parking in Houston. Since the predicted $A0$ and $A14$ values sum to zero, the estimated turnaround and block time duration from the PTFM are the corresponding values of turnaround duration and block time retrieved from the *disrupted* flight capacity and *disrupted* flight service management ANNs. As such, the PTFM estimates of turnaround duration and block time are $38.37mins$ and $93.61mins$, respectively, for the disrupted flight from Dallas to Houston.

The actual observed turnaround duration and block time for the disrupted DAL-HOU flight are $38mins$ and $104mins$ respectively. This indicates a 0.97% difference between the PTFM estimate for turnaround duration and the actual turnaround duration of the weather-disrupted DAL-HOU flight. Similarly, there is a 10.52% difference between the PTFM estimate for block time and the actual observed block time of the weather-disrupted flight from Dallas to Houston. The execution of the disrupted flight schedule from Dallas to Hous-

ton resulted in a tactical delay of $3mins$ during turnaround at Dallas and a strategic delay (i.e. discretionary holdup by human specialists in the AOCC) of $39mins$ from aircraft pushback at Dallas to aircraft parking at the arrival gate in Houston. On the other hand, the PTFM (i.e. system of ANNs) estimated a tactical delay of $8.34mins$ and a strategic delay of $1.59mins$ for this particular disrupted flight schedule from Dallas to Houston. As such, there is a 94.18% difference between the tactical delay applied to resolve the disrupted DAL-HOU flight during schedule execution when compared to the tactical delay estimate from the PTFM. Similarly, the percentage difference between the strategic delay applied to resolve the disrupted flight during schedule execution and the strategic delay estimated by the PTFM is about 180%.

Chapter Summary: This chapter provided a detailed review of the supervised and unsupervised techniques that enabled the creation of two separate platforms for modeling intelligent agents for predictive purposes in a simultaneously-integrated recovery paradigm for airline disruption management. By applying a fusion of abstraction techniques through exploratory data analysis (discussed in Chapter 3) and domain knowledge of airline operations control, we developed and evaluated the frameworks for the uncertainty and predictive transfer function models (i.e. UTFM and PTFM) that constitute a generic intelligent agent in the AOCC for airline disruption management. In the next chapter, we elucidate the tenets for the integration and interaction of multiple intelligent agents in the AOCC that reveal simultaneously-integrated recovery during airline disruption management.

5. ENABLING INTEGRATION AND INTERACTION OF INTELLIGENT AGENTS FOR A *SIR* PARADIGM IN AIRLINE DISRUPTION MANAGEMENT

Humans assume a primary role in the development, evolution, and assessment of existing paradigms and systems adopted for making decisions during airline operations recovery and disruption management [48]. Current paradigms for operations recovery in many airlines involve database query systems (DBQS), which allow human operators (or specialists) in the AOCC to perform inquires on databases in order to effectively assess solutions proffered by decision support systems (DSS) for different problem dimensions (i.e. aircraft, crew and passenger) during irregular operations and disruption management. Together, DBQS and DSS provide a credible platform

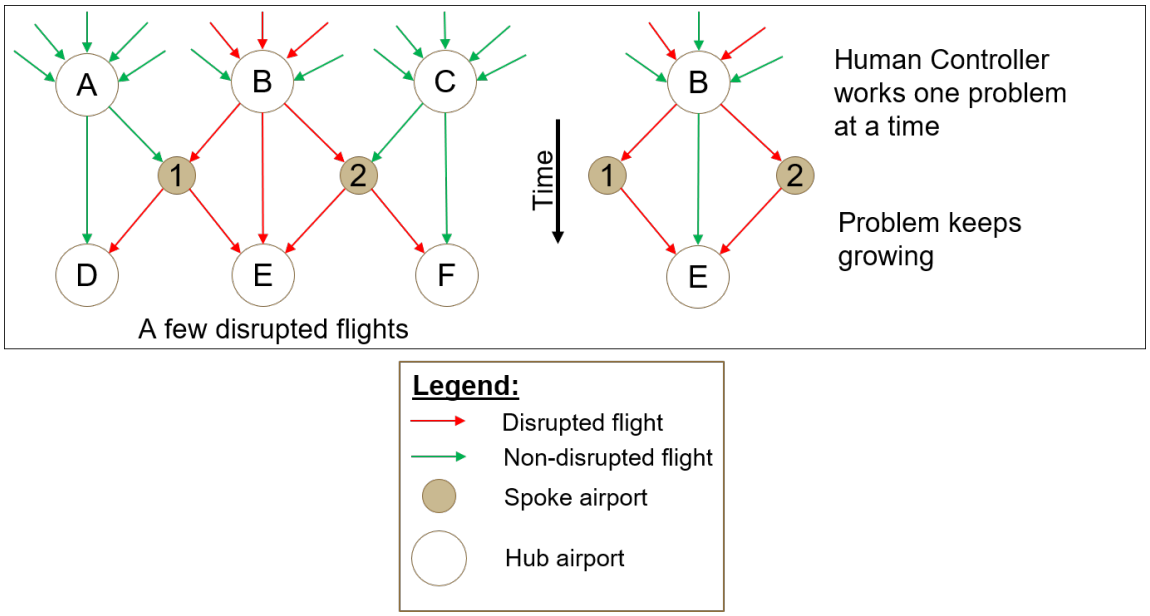

Figure 5.1. Human involvement in disruption management and operations recovery at the AOCC

for addressing irregular operations, as shown in Fig. 5.1, during schedule execution on day of operation. However, the efficiency of this platform is bottlenecked by human operators for two reasons. First, humans find it difficult to simultaneously utilize large volumes of data to make the best decision that spans multiple problem dimensions for disruption management [5,138]. In order words, they can not promptly employ all available and necessary data to select and apply appropriate information that is most effective for managing irregular operations. Second, plausibly inferior quality (i.e. goodness) of information (retrieved by the human operator from the database query system) that is subsequently passed on to the decision support system can compromise the efficacy of solutions provided by the decision support system, thus rendering disruption resolutions sub-optimal. To complicate decision-making matters, existing paradigms for airline disruption management do not readily allow human experts to see the impact of these disruption resolutions (partly informed by DBQS) until the recovery plan is already in motion and its consequence can not be revoked [48]. As such, delineating skill from luck while managing disruptions for optimal airline recovery remains an open and challenging problem. To this effect, current industry techniques for airline disruption management are unable to attain and maintain expeditious, effective and concurrent recovery of all problem dimensions in a disrupted airline route network.

Furthermore, many decision support systems adopted by the current integrated recovery paradigms for airline disruption management are imbued in a monolithic system design doctrine, wherein specifications are created first before a system that meets the specifications is constructed. However, this design approach has not succeeded in eradicating irregular airline operations because specifications continue to evolve as new capabilities are added to an existing system [47]. Although current approaches for airline disruption management are capable of providing decision support, albeit restricted by a sequential process resolution of problem dimensions, the rapid evolutionary manner in which different situations for irregular operations occur typically renders DSS solutions ineffectual within moments after they are generated.

Moreover, adding more proficiency, in form of computerized information systems, to existing decision support systems in the AOCC to address fleeting solution validity significantly increases the complexity of the monolithic and centralized design approach for airline disruption management. To this end, there is a need to substantially augment the decision-making capacity of human specialists in the AOCC with modular and decentralized decision support platforms that allow real-time generation and tracking of disruption resolutions with minimal complexity during irregular operations and schedule recovery.

Recent advancements in artificial intelligence (AI) and distributed ledger technology (DLT) [37, 129, 139–142] have provided an avenue to develop decision support systems that can allow human specialists in the AOCC to readily assess and validate the effectiveness of their decisions while concurrently recovering the airline network during irregular operations. To that effect, this chapter provides a compendious discussion of the mechanisms that enable the integration of constituent AI models (i.e. UTFM and PTFM elucidated in Chapter 4) that define the intelligent agent for each functional domain in the AOCC, and the interaction of multiple intelligent agents for simultaneously-integrated recovery during airline disruption management.

We begin this chapter by elucidating different tenets of artificial intelligence and distributed ledger technology, which provide a conducive medium for achieving scalable simultaneously-integrated recovery of airline schedule during disruption management. Next, we describe the consensus platform for a special type of distributed ledger called Hashgraph and how routines that define the UTFM and PTFM (i.e. AI agents) serve as elements for a proof-of-stake system for achieving simultaneously-integrated recovery. We conclude the chapter with the discussion of a demonstration of the AI-DLT synthesis for simultaneously-integrated recovery of sets of disrupted flight schedules across multiple functional domains in the AOCC.

5.1 A Symbiotic Synthesis of AI and DLT

One of the primary objectives of the fourth industrial revolution (i.e. Industry 4.0) is to transform traditional industrial practices by combining these practices with the latest smart technology [143]. As such, current industry practices for irregular airline operations, which involve multiple air transportation stakeholders, can benefit amply from the use of machine to machine communication and internet of things (IoT) deployments to achieve increased automation, better communication, and self-monitoring during disruption management without significant need for human intervention. Furthermore, the unrelenting ubiquity of data that span many different areas of society has necessitated and enabled the creation of new interdisciplinary principles founded on mathematics, statistics, and probability theory, which enable machines (or computers) to have cognitive functions to learn, infer and adapt by leveraging data [144, 145]. To this effect, there is a strong mandate to explore decentralized interactions among intelligent machines that represent functional roles in the AOCC for better disruption management.

5.1.1 Artificial Intelligence

Artificial intelligence or AI is the field that studies and applies interdisciplinary principles to achieve computationally intelligent agents that can directly assist humans with their day-to-day functions. A very prominent principle used in AI is machine learning [33, 42], which relies on a centralized model for training wherein a group of servers run a particular algorithm against many training and validation examples, as described in Chapters 3 and 4. As such, the few AI systems that exist (or in development) today for airline disruption management are generally specialized expert systems that utilize rolling and centralized data from a DBQS to assist human specialists in the AOCC with making decisions during irregular operations [120, 146]. Thus, current AI trends for airline disruption management aim to enable automated machine learning processes to manage data acquisition and knowledge updates for

database query systems (DBQS), thereby minimizing the manual labor required by human specialists for developing and evaluating decision support systems to mitigate irregular operations during schedule execution. Furthermore, the complex nature of the interaction between different actors (e.g. functional roles in the AOCC and air transportation stakeholders like air traffic controllers) does not augur well for robust airline disruption management, because a centralized AI system can not efficiently capture the proclivities amongst actors to discover and track emergent behavior in collective decision-making during schedule execution in a scalable manner [87, 147, 148]. As such, a complementary platform is required to enhance the performance qualities of AI systems for improved disruption management during irregular operations.

5.1.2 Distributed Ledger Technology and Blockchain

Distributed ledger technology or DLT [149] represents an emerging technology that embodies two peculiar properties. First, it is distributed in nature such that the agreement about the state of a particular ledger being maintained is attained through recompensed consensus by a network of intelligent agents in lieu of relying on trust in a third-party intermediary that is extraneous to the network. Second, intelligent agents can deposit digital assets such as acts, timestamps, and states in the ledger, whose records are readily auditable, transparent, and incommutable based upon cryptographic and distributed basis that resist censorship and manipulation of assets [150]. To that effect, DLT enables high levels of anonymity and pseudonymity for intelligent agents during transactions (i.e. digital asset trades), such that records of transaction activities are observable at the meta-level and resistant to manipulation. However, the identification of specific intelligent agents performing trades on the ledger remains impossible during transactions, thereby rendering DLT immensely resilient to the intolerance of false information sharing for enhanced data security. The first noteworthy application of DLT was for the cryptocurrency named Bitcoin [151, 152].

The blockchain, which is a rudimentary but effective form of DLT, was used to solve the problem of distributed consensus in a trustless network that provides a secure, controlled, and decentralized method of minting the Bitcoin digital currency. At its core, blockchain is a layer of a distributed peer-to-peer network running on the front end of the internet. In that regard, blockchain is characterized by a data structure for tracking digital footprints in logical blocks of information, a shared ledger for recording actions (e.g. digital currency mining activities) by member computer systems or intelligent agents, and a decentralized consensus mechanism that solves a complex and random mathematical problem to validate member interactions based upon different routines that define engagement rules for participating members. DLT can be either permissioned or permissionless depending on the manner in which access for information exchange is granted to current and prospective members of the DLT. Thus, a permissioned DLT only allows authorized intelligent agents to access the DLT application in private, consortium, or cloud-based settings, while a permissionless DLT like the Bitcoin blockchain is publicly accessible to any intelligent (mining) agent via the internet.

5.1.3 Decentralized AI for Airline Disruption Management

Background

The existing scope of AI applications for airline disruption management primarily focuses on automated machine learning to improve the quality of information retrieved by human specialists from database query systems (DBQS) [146]. While AI provides a means for improving the decision-making flexibility of human specialists in the AOCC during irregular operations, the centralized nature of existing monolithic systems for disruption managment does not enhance the efficiency of the decision-making process as a whole [20, 153]. Thus, agile decision-making for irregular operations during airline schedule execution requires a platform that inculcates decentralized intelligence during disruption management [154].

Furthermore, many existing integrated recovery paradigms for airline disruption management employ decision support mechanisms (i.e. time-space optimization formulations) that provide disruption resolutions for separate problem dimensions without readily disclosing the unique rationale behind the solution generation process for a specific disruption event. As such, human specialists in the AOCC are often detached from the resolution creation process of the DSS. In addition, majority of existing system design paradigms for airline operations recovery are *Models and Algorithms (MALG)* typically created by developers that operate separately from the AOCC organization, and as such, *Models and Algorithms* are not willingly accepted nor included in many highly customized tools and systems currently used by the AOCC for disruption management [9, 153, 155].

A recent study of the Southwest Airlines Baker workgroup for disruption management, conducted by Deloitte Insights [48], revealed that having superintendents of dispatch (i.e. human specialists in the AOCC) manage the creation of DSS that they themselves would use guaranteed that the disruption resolution sought by developers and users was shared and agreeable. This way, human specialists from different functional roles in the AOCC were more likely to accept and commit to a disruption resolution outcome proferred by a decision support system that is truly representative of their individual and collective decision-making expertise. As such, human specialists in the AOCC would prefer a DSS that can expediently verify the efficacy and validity of their decision to implement a certain disruption resolution, in lieu of extant DSS that propose a disruption resolution to which the specialists have to decide whether or not to implement the proffered resolution. To that effect, a decentralized AI platform lends a suitable medium for developing next generation DSS for airline disruption management.

Taxonomy

The synthesis of artificial intelligence and distributed ledger technology (i.e. decentralized AI) can be categorized into two separate but related decentralized platforms, based upon certain integration properties and benefits, namely: *Blockchain for AI* and *AI for Blockchain* [156, 157]. *Blockchain for AI* represents a disruptive integration of artificial intelligence and distributed ledger technology that aims to help AI systems to attain the following enhancements: i) a secure data sharing environment for intelligent agents, ii) decentralized computing for intelligent agents, iii) explainable rationale for the actions of intelligent agents, iv) the coordination of distrusting intelligent agents. In complement, *AI for Blockchain* represents an integration of artificial intelligence and distributed ledger technology that seeks to use AI to improve the automation, decision-making, and optimization of DLT (such as blockchain) for enhanced performance and governance. In that regard, *AI for Blockchain* aims to achieve the following enhancements for distributed ledger technology: i) secure and scalable distributed ledgers, ii) readily customizable distributed ledger systems that preserve the privacy of intelligent agents, iii) automated referee-

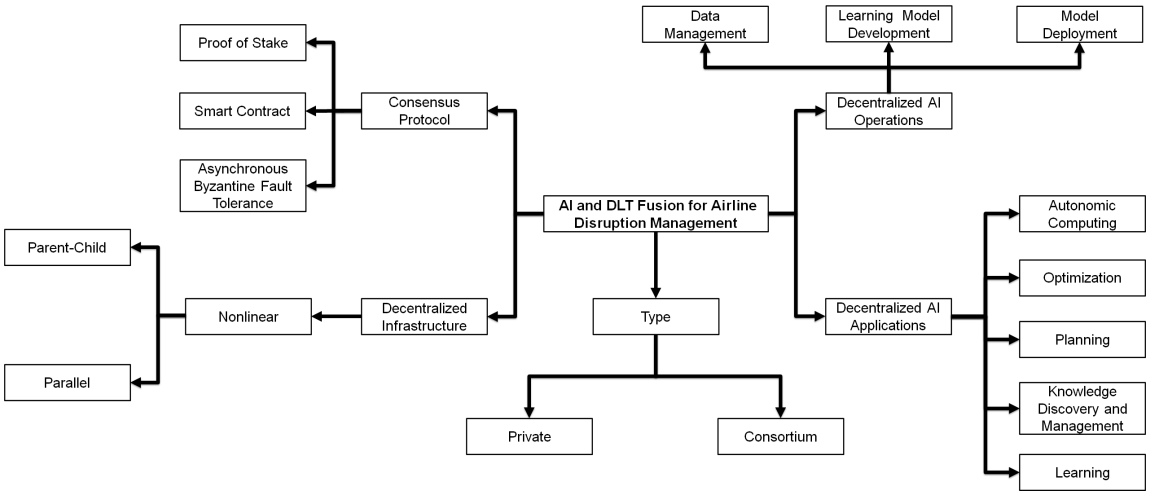

Figure 5.2. Taxonomy of a decentralized AI platform for airline disruption management

ing and governance of participating intelligent agents for distributed computing. A hybrid of these platforms represents a unique platform that enables multiple separate enhancements applicable to *Blockchain for AI* and *AI for Blockchain*. Fig. 5.2 shows a tree-diagram that represents five separate aspects of taxonomy in decentralized AI for airline disruption management based upon type, application, operation, consensus, and infrastructure [158].

- **Type**: The type of decentralized AI platform for airline disruption management is defined by the caliber (i.e. importance and hierarchy) of stakeholders involved in irregular operations. As such, there are two appropriate types of decentralized AI namely: private and consortium AI systems respectively. A private decentralized AI for airline disruption management represents a *Blockchain for AI* platform that models the intrinsic behavioral properties of individual high-level stakeholders (i.e. airlines, airports, policy makers, etc) in the air transportation network during disruption management. Thus, a private decentralized AI for airline disruption management models the interaction amongst functional roles (i.e. low-level actors) within high-level stakeholder organizations during schedule execution and irregular operations. To this effect, a private decentralized AI presents a credible avenue for a system of systems modeling architecture for the AOCC [159, 160]. In complement, a consortium decentralized AI for airline disruption management represents a hybrid of the *Blockchain for AI* and *AI for Blockchain* platforms that models the interaction amongst multiple high-level stakeholders within the air transportation system during disruption management. As such, a consortium decentralized AI provides an appropriate means for modeling irregular operations in the air transportation network as a federated system of systems [161].

- **Application**: The application of a decentralized AI platform for airline disruption management allows for the following enhancements to existing decision support systems:

1. *Autonomic computing* - This ensures that intelligent agents, such as functional roles in the AOCC, are able to cope when subjected to heterogeneity at all verticals including data sources, data processing, and data storage in order to perceive their internal states and conduct specified actions accordingly. As such, distributed ledger technology (e.g. blockchain) provides a medium to permanently track the interactions among multiple intelligent agents across separate levels of hierarchy within the air transportation system.

2. *Optimization* - Existing paradigms for airline disruption management are enforced with centralized optimization strategies that impose global (i.e. system-wide) objectives such as maximizing airline profit margin, which result in strictly constrained and subordinate system-wide behaviors and performance to insure local objectives (such as minimizing flight delay duration) of constituent actors [20, 109]. Thus, the application of decentralized AI strategies through distributed ledger technology (i.e. *Blockchain for AI*) creates a distributed optimization platform that concurrently increases the search and design space to allow for improved system-wide performance for constrained and unconstrained interaction of intelligent agents in the AOCC.

3. *Planning* - From an airline stakeholder's perspective, the outcome of the integration of artificial intelligence and distributed ledger technology during schedule execution is to readily attain a dynamic schedule planning scheme while managing airline disruptions. Existing integrated recovery paradigms are limited to static and tedious re-planning routines that take considerable amount of time (due to significant human interventions) to obtain an updated planning scheme. In that regard, *Blockchain for AI* provides an applicable platform to obtain new planning strategies that utilize decentralized and distributed optimization routines for recording multi-

ple planning schemes and their respective evolution histories (i.e. provenance) [158].

4. *Knowledge discovery and management* - The management of data for separate intelligent agents in existing systems is centralized, and as such, creates a monolithic system that can not enable peer-to-peer interaction amongst system constituents for disruption management. As such, the decentralized *Blockchain for AI* platform provides a means for unstructured peer-to-peer network interaction of low-level stakeholders that represent functional roles or domains in the AOCC of an airline. Hence, emergent behavior in form of new sequence of resolution activities amongst multiple functional roles can be discovered during disruption management. In complement, *AI for Blockchain* provides a medium for a structured peer-to-peer network interaction of high-level stakeholders (created via *Blockchain for AI* platforms) that represent primary actors in the air transportation network, such as multiple airlines and airports. As such, emergent behavior in terms of different collaborative decision-making (CDM) [162, 163] routines for airline disruption management can be enabled through autonomic AI computing.

5. *Learning* - Existing AI platforms for airline disruption management are strictly trained and employed by using a centralized framework to achieve global intelligence. However, decentralized AI platforms are capable of autonomous and immutable learning through distributed computing that promotes fully coordinated local intelligence to achieve new calibration routines for global intelligence. For instance, the sequence of real-time events to re-plan a disrupted airline network may require the interaction many functional roles (i.e. intelligent agents) in the AOCC. As such, the transactions amongst functional roles seeking consensus on a distributed ledger creates a platform to discover new learning routines for different scenarios of airline disruption.

- **Operation**: The operation of a decentralized AI platform for airline disruption management represents its capacity to readily manage copious amounts of data for adaptable decision-making through AI applications. Unlike centralized AI systems with strict system specifications and operations, decentralized AI operations ensure versatility in data for separate functional roles in the AOCC through the following means:

  1. *Data management*: There is a high proclivity for data duplication in existing DSS because small changes in data content typically results in repeated transfer of updated datasets during disruption management [2, 164, 165]. As such, centralized data management becomes significantly inefficient as the airline route network (or air transportation network) expands. This inefficiency is characterized by rapid bandwidth overloading experienced by human specialists and increased backhaul network traffic that create substantial latency issues in existing DSS. Thus, decentralized data management provides an avenue for explicit DSS modeling routines to be simultaneously developed and deployed at node levels that represent fundamental components of intelligent agents in a multi-agent system. Consequently, these DSS modeling routines serve as dynamic metadata that can provide substantial latent information to a distributed ledger platform while tracking and maintaining provenance and immutability of disruption resolutions during consensus.

  2. *Learning model development* - Centralized learning creates a platform where learning models are calibrated and tested before software deployment [138]. As such, centralized platforms are unable to accommodate for rapidly evolving data streams because of the dynamic and cascading nature of the impact of airline disruptions on the air transportation network. In that regard, decentralized AI platforms, especially *Blockchain for AI*, enable the premeditation and predetermination of all possible search rules for addressing a specific type of disruption from separate AI agents. These

search rules are made readily accessible to a consensus algorithm based upon the engagement routines (i.e. smart contracts) that activate the instantiation of a specific search rule during negotiations amongst intelligent agents.

3. *Model deployment* - Complementary to learning model development, AI replaces the brute force approach present in many centralized learning platforms that first solve an explicit optimization problem by trying every possible combination of decision variables, before recommending suitable solutions for the recovery of the airline network. Hence, distributed ledger technology adopts different engagement routines to qualitatively and visually reveal the interaction of various search rules employed (i.e. learned and cached a priori from data) by multiple AI agents for real-time recovery of a disrupted airline network.

- **Consensus**: The consensus of a decentralized AI platform for airline disruption management relies on the capacity of multiple participating functional roles in the AOCC to agree on a set of acts for recovering a disrupted route network [153]. As such, the following artifacts provide pertinent properties for enforcing consensus in a decentralized multi-agent system for airline operations recovery.

  1. *Byzantine fault tolerance (BFT)* - This represents a majority voting algorithm that eliminates transaction validation from malicious participants on a distributed ledger [39, 139]. Malicious participants represent intelligent agents (or functional roles) in a permissioned DLT platform (i.e. multi-agent system) that can directly or indirectly manipulate the outcome of a recovery plan for a disrupted airline network. As such, many existing BFT algorithms guarantee consistent maximum fidelity and operability of a multi-agent system if at least two-thirds of all participants in the decentralized AI platform are not malicious [158].

2. *Proof of stake (PoS)* - This represents a type of routine that selects stakeholders (e.g. functional roles in the AOCC) as signatories and validators of disruption events and corresponding disruption resolutions, respectively, during irregular operations [149]. To that effect, stake in PoS indicates the degree to which the efficacy of the decisions (or resolutions) of a stakeholder can be trusted, or the measure of stakeholder interest in the effectiveness of a decentralized AI interaction for airline disruption management. A typical airline route network is somewhat delay-tolerant due to the finite capacity for useful resources in the air transportation network [166]. Hence, PoS provides an energy-efficient protocol for unconstrained interactions amongst stakeholders as long as stakeholders are defined by appropriately calibrated AI systems embedded with possible and readily applicable search rules for different disruption resolutions.
3. *Smart contract* - This represents computational models (i.e. computer programs) that execute a set of engagements specified in digital form [167]. To this effect, constituent AI models for intelligent agents, such as functional roles in the AOCC, define the terms, conditions, and execution of a smart contract for airline disruption management. Furthermore, smart contracts assimilate and execute workflows, and as such, present a viable means for automating decision-making for regulatory compliance and approbation of several air transportation and flight operation processes during airline schedule execution.

- **Infrastructure**: The infrastructure of a decentralized AI platform for airline disruption management represents the architecture (or environment) in which the interaction amongst separate stakeholders acting on a distributed ledger is simulated for any scenario of irregular operations. Many DLTs such as the Bitcoin blockchain are based upon a linear infrastructure wherein blocks of data content are sequentially connected via hashing mechanisms for a single chain link to achieve immutability and transparency. As such, single-chained DLTs

(i.e. linear blockchain infrastructures) do not scale up well and struggle to achieve acceptable real-time performance necessary for effective decentralized applications [168]. Furthermore, to enable concurrent and independent interactions of multiple intelligent agents in a multi-agent system, distinct single chains are necessary to record the individual transaction activities of each intelligent agent, thereby rendering digital exchanges of assets, value, and information impossible for linear blockchain infrastructurs in heterogeneous environments. To this end, nonlinear DLT infrastructures based upon graph theory provide a fitting platform for developing and deploying decentralized AI frameworks for airline disruption management that promote scalable real-time applications while utilizing big data.

Many existing nonlinear DLT infrastructures exist in the form of a directed acyclic graph (DAG) [169–171] that create multi-chain architectures such as parent-child chains and parallel chains. Parent-child chains represent multi-chain architectures where one or more chains serve as the primary chain that records the information and digital activities of other chains. As such, from a high-level perspective, a parent-child chain presents an appropriate nonlinear infrastructure for interaction in a consortium decentralized AI for airline disruption management, such that the primary chain is representative of a major policy maker (e.g. FAA) in the air transportation system while other chains in the consortium represent stakeholders like airlines and airports that are subordinate to the compliance directives (defined by smart contracts) provided by the policy maker. In transition, parallel chains represent multi-chain architectures where constituent chains operate independently from other chains. Thus, the conscription of the rules of engagement for performing a transaction on a specific chain is controlled by a particular intelligent agent, and is sovereign from the actions of other intelligent agents operating on other chains in the decentralized platform. Hence, from a low-level perspective, a parallel multi-chain architecture provides a fitting nonlinear infrastructure for interaction in a

private decentralized AI for airline disruption management, wherein each chain is representative of queuing disruption and resolution information from a particular functional role in the AOCC during airline schedule execution.

5.2 A Case Study

For an exhibition of the fusion of artificial intelligence and distributed ledger technology for airline disruption management, we consider a system of systems modeling of the network operations control center at Southwest Airlines (i.e. SWA-NOC). For our demonstration, each of the eleven functional roles in SWA-NOC is modeled as a separate AI system (i.e. intelligent agent) defined by uncertainty and predictive transfer function models (UTFM and PTFM) described in Chapter 4.

The uncertainty transfer function model or UTFM provides the most likely sequence and likelihood of decision-making activities at different phases of flight for a specific resolution during disruption management. Thus, from an information theory perspective [172, 173], the UTFM estimates the maximum amount of information or recovery uncertainty (measured as negative log likelihood) that can be predicted for a specific disruption resolution considered by a human specialist in a functional role

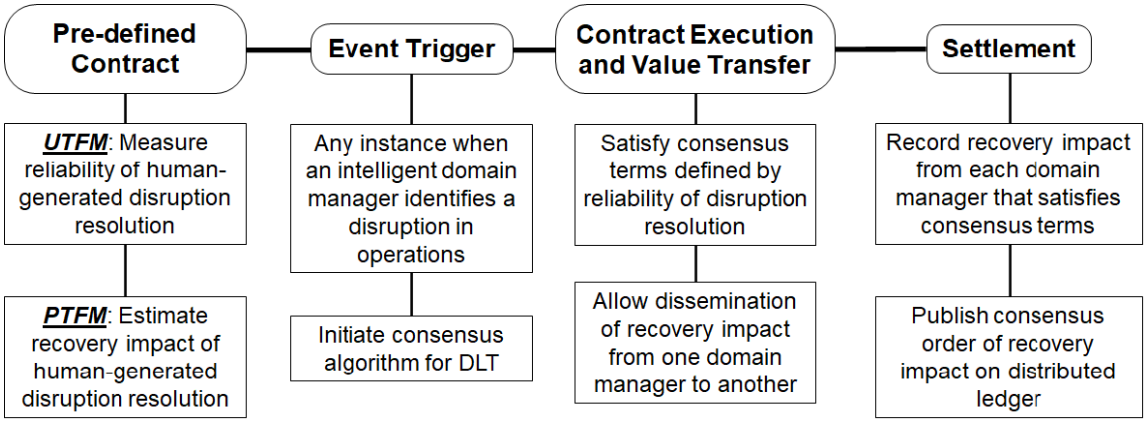

Figure 5.3. Integration and interaction routine in a decentralized AI platform for airline disruption management

during airline operations recovery. The recovery uncertainty estimated by the UTFM represents the reliability of the disruption resolution provided by the human specialist, which is subsequently used to define the stake of a specific functional role during its interaction with other functional roles in SWA-NOC to achieve consensus during disruption management. In complement, the predictive transfer function model or PTFM provides an evaluation of the required duration and delay at different phases of flight upon enacting a particular resolution during disruption management. As such, from an airline operations recovery perspective [15, 20, 174, 175], the PTFM estimates the recovery impact for a specific disruption resolution conceived by a human specialist in a functional role during airline disruption management. Thus, for any functional role (i.e. intelligent domain manager) operating based upon the routine defined in Fig 5.3 for a private decentralized AI platform, the UTFM and PTFM conjointly define a smart contract that measures the reliability of a human-generated disruption resolution and its corresponding effect on the recovery plan for a disrupted airline route network, thereby enabling integration in a multi-agent system for airline disruption management. Furthermore, the execution of the smart contract on a DLT is instantiated by the occurrence of disruptions that induce irregular operations for specific functional roles in SWA-NOC, thereby enabling interaction in a multi-agent system for airline disruption management. As such, the multi-agent system for airline disruption management investigated in this case study represents a hybrid decentralized AI platform that enables multiple separate properties of the *AI for Blockchain* and *Blockchain for AI* platforms for integration and interaction respectively. To this end, the subsequent parts of this section describe relevant rationale used to calibrate AI models as intelligent agents for integration into an applicable DLT protocol that enables creation and observation of interactions amongst intelligent agents during airline disruption management.

5.2.1 AI Calibration

Artificial intelligence calibration is the process of adapting the behavior of an intelligent agent in a multi-agent system for a specific research scope and environment in order to attain high fidelity real-world simulations [46,176]. As such, to foster trust in airline network recovery obtained through human-AI collaboration, the estimation of the reliability of human-generated disruption resolutions by an intelligent agent during airline disruption management must be acceptable [177,178]. To this effect, we discuss the rationale used to calibrate the UTFM for intelligent agents that represent functional roles in SWA-NOC.

Definition

The UTFM, detailed in Chapter 4, represents a design platform for creating probabilistic finite state machines [179, 180] that define the manner in which disruption resolution activities, i.e. a series of input criteria provided by a human specialist, are translated by employing probabilities to evaluate the chance with which the disruption resolution yields the outcome intended by the human specialist. Hence, for a human specialist capable of providing a set of possible disruption resolutions (i.e. outcomes) $r_1, ..., r_k$, let X represent a random variable that represents the unpredictable outcome of a single disruption resolution trial. As such, there exist values $p_1, ..., p_k \in [0, 1]$ wherein

$$p_i = Pr[X = r_i] \tag{5.1}$$

and

$$\sum_{i=1}^{k} p_i = 1 \tag{5.2}$$

Note that the term $Pr[*]$ means " the probability or likelihood of $*$ ".

Thus, for every state $s \in S$ in a probabilistic finite state machine and every action or symbol $a \in A$ defined by a specific disruption resolution, the following expressions are valid:

$$\begin{aligned} &either \quad \Delta(s,a) = \emptyset \\ or \quad &\sum_{s' \in \Delta(s,a)} Pr[s \xrightarrow{a} s'] = 1 \end{aligned} \tag{5.3}$$

Hence, for any series of input criteria $x \in A^*$ of length k and every trace t that defines the probabilistic finite state machine in the form:

$$t = s_0, x_1, s_1, ..., s_{k-1}, x_k, s_k \tag{5.4}$$

the probability that the trace defined by Eqn 5.4 represents the path taken through the finite state machine upon reading the input criteria x is expressed as:

$$Pr[t] = Pr[s_0 \xrightarrow{x_1} s_1] \quad ... \quad Pr[s_{k-1} \xrightarrow{x_k} s_k] \tag{5.5}$$

Congruently, for all scenarios where $s' \notin \Delta(s,a)$,

$$Pr[s \xrightarrow{a} s'] = 0 \tag{5.6}$$

To this end, the probability that a disruption resolution provided by a human specialist is accepted by an aggregate probabilistic finite state machine F that represents the UTFM, is based upon a set of traces $T(x)$ for x wherein the final state s_k of each trace is an accept state and $Pr[t]$ is summed over all such traces t by the following expression:

$$Pr[x \quad \text{is accepted}] = \sum_{t \in T(x), s_k \in F} Pr[t] \tag{5.7}$$

Learning Probabilities for Integration in SWA-NOC

As we detailed in Chapter 4, the real world application of probabilistic finite state machines for airline disruption management is made possible through a special state machine variant called the hidden Markov model. As such, the UTFM is representative of a hidden Markov model (i.e. a transducer-style probabilistic finite

state machine) wherein outcomes are randomly generated at a state (in lieu of along transitions) according to a random stochastic process that models the activities at that state [181]. Thus, for each state $s \in Q$ of the hidden Markov model, there exists a random variable associated with s that takes in values of disruption resolution x according to certain state-dependent probabilities. As such, it is imperative that these state-dependent probabilities are appropriately representative of the frequency to which the random process (or variable) associated with a particular state is instantiated during disruption management in the real world. To this effect, inferring the structure (i.e. state transition and observation probabilities) of a hidden Markov model from real world data presents a common learning problem in artificial intelligence.

A variant of the expectation maximization algorithm [182, 183] called the Baum-Welch algorithm [184], described in Chapter 4, provides a credible platform for learning the probabilities of a hidden Markov model from data through an unsupervised optimization routine that guarantees local convergence but not global convergence. As such, the Baum-Welch algorithm seeks to facilitate optimal prediction on training data that represents empirical frequency. However, empirical frequency does not provide a good estimate of the probability of new real-world situations. And just because an intelligent agent, defined by a model based upon empirical frequency, has not observed some value of a variable does not mean that the likelihood that the value exists should be zero (i.e., an impossible value does not exist) [185]. Thus, to ensure high quality and fidelity of constituent UTFM structures for an intelligent agent, we employ prior domain knowledge on airline scheduling and operations to solve the zero-probability problem by implementing pseudocount (or prior count) for each value to which real-world data values are added during Baum-Welch training.

To understand the pseudocount effect, suppose there is a binary feature Y and a training data set with n_0 observed instances where $Y = 0$ and n_1 instances where $Y = 1$, then a pseudocount $c_0 \geq 0$ is defined for $Y = 0$ and another pseudocount

$c_1 \geq 0$ for $Y = 1$ so as to estimate the probability (or inherent likelihood) of an outcome as:

$$Pr[Y = 1] = \frac{n_1 + c_1}{n_0 + c_0 + n_1 + c_1} \tag{5.8}$$

Without loss of specificity, suppose Y is defined by the domain $y_1, ..., y_k$, then a pseudocount c_i is defined for each y_i such that all prior counts are selected before training commences. As such, given a training data set with observed real-world examples where n_i represents the number of instances with $Y = y_i$, then:

$$Pr[Y = y_i] = \frac{n_i + c_i}{\sum_i n_i + c_i} \tag{5.9}$$

Thus, choosing pseudocounts represents a supplementary part of designing the UTFM learner that effectively estimates how much an intelligent agent should accept (or believe) an input criterion y_i if it had seen one instance with y_i to be true during Baum-Welch training as compared to if it had seen no instances with y_i to be true. As such, if there were no instances of y_i to be true during training, then the intelligent agent believes that y_i is impossible and c_i was set as zero during training. However, setting c_i as zero during Baum-Welch training of data sets with substantially small n_i may cause the optimization to diverge, and not arrive at a local optimal due to the undefined nature of the logarithm for a value of zero. Therefore, prior counts provide a means to avoid to this problem. If there are instances with y_i to be true during training, then the intelligent agent accepts that a value for y_i observed one time would be $\frac{1+c_i}{c_i}$ times more likely than a value observed zero times. Thus, the prior knowledge factor, f_{pk}, for the consideration of a specific value in a data set during Baum-Welch training of constituent probabilistic finite state machines for the UTFM is expressed as:

$$f_{pk} = \frac{1 + c_i}{c_i} \tag{5.10}$$

For example, we use the binary feature that indicates a flight swap from the *non-disrupted* data set to fix pseudocounts for Baum-Welch training of constituent probabilistic finite state machines for the UTFM, which define the random processes

for remaining in the tactical and strategic (i.e. Schedule and Outcome) phases of activity (refer to Fig. 4.4) during disruption management. As mentioned in Chapter 4, the *non-disrupted* data set represents approximately 620,000 instances of Southwest Airlines flight schedules that executed without disruptions between September 2016 and September 2017. About 16,160 flight schedules in the *non-disrupted* data set executed with a flight swap. The subset of flight schedules that executed with a flight swap were neither driven by the occurrence of active disruptions nor required irregular operations (i.e. considerable AOCC intervention) for execution. Theoretically, from an airline disruption management perspective, swapping a flight prior to schedule execution that is not necessitated by the existence of a disruption should be impossible. As such, the total number of instances of these flight schedules in the *non-disrupted* data set represents an appropriate pseudocount value for eliminating the zero-probability problem during Baum-Welch training of tactical and strategic probabilistic finite state machines for the UTFM. Thus, there is at least a 2.61% probability (i.e. inherent likelihood) that any value for remaining in the tactical and strategic phases of activity that is observed once would be 0.0062% more likely than any value that is not observed at all during Baum-Welch training.

The arrival of a flight at the exact time planned prior to schedule execution is almost impossible when there are disruptions in the airline route network [186]. As such, we use the binary feature that indicates precise on-time arrival (i.e. A0) from the *disrupted* data set to set pseudocounts for Baum-Welch training of probabilistic finite state machines for the UTFM that define the random processes for remaining in the operational (i.e. Outcome) phase of activity during airline disruption management. The *disrupted* data set contains about 434,000 instances of Southwest Airlines flight schedules that were subject to flight delays across multiple functional roles in the AOCC between September 2016 and September 2017. In that regard, we use the binary features that indicate route origination (i.e. first scheduled flight on day of operation) and flight swap from the *disrupted* data set to specify pseudocounts for

Table 5.1. Pseudocount and inherent likelihood of observing new values for all functional roles in Southwest Airlines network operations control

Functional Domain	**Tactical Pseudo-count**	**Operational Pseudo-count**	**Strategic Pseudo-count**	**Inherent Tactical Likelihood**	**Inherent Operational Likelihood**	**Inherent Strategic Likelihood**
Customer Hold	1,906	11,289	3,222	0.041	0.243	0.069
Dispatch CSC	1,988	3,160	4,792	0.115	0.183	0.277
Flight Operations	6,365	10,580	2,682	0.177	0.294	0.074
Fuel Management	603	1,180	228	0.126	0.246	0.048
Ground Operations	8,146	48,827	5,748	0.049	0.293	0.034
Inflight	4,751	25,423	3,505	0.060	0.323	0.045
Maintenance	6,985	4,901	4,648	0.211	0.148	0.140
NAS	2,221	2,774	1,184	0.099	0.124	0.053
Security	850	955	145	0.291	0.326	0.050
Technology	869	2,206	397	0.098	0.249	0.045
Weather	1,483	597	1,065	0.118	0.048	0.085

Baum-Welch training of probabilistic finite state machines that define the transitions along and across different phases of activity in the UTFM respectively.

Table 5.1 reveals the pseudocount and inherent likelihood of observing new values (i.e. disruption resolution criteria) for each functional role in Southwest Airlines network operations control, based upon an assessment of a training subset of the *disrupted* data set that represents 99% of all instances of flight schedule execution in the *disrupted* data set. Note that the remaining 1% represent instances of flight schedules in the *disrupted* data set that are used later on in this chapter as new (disrupted) flight schedules for demonstrating interaction in the decentralized AI platform. As shown in Table 5.1, the Security functional role has the highest inherent likelihoods of 29.1% and 32.6% for observing new values for tactical and operational disruption management respectively. Similarly, the Dispatch CSC functional role has the highest inherent likelihood of 27.7% for observing a new value for strategic disruption management. Conversely, the lowest inherent likelihoods of observing a new value for tactical, operational, and strategic disruption management are 4.1%, 4.8%, and 3.4% respectively, and are realized by the Customer Hold, Weather, and Ground Operations functional roles. Thus, human specialists in the Security and Dispatch CSC functional roles have the most trial-and-error learning tendencies during different phases of airline disruption management. In contrast, human specialists in the Weather, Customer Hold, and Ground Operations functional roles have the least trial-and-error learning proclivities during separate phases of airline disruption management.

5.2.2 DLT Protocol

Distributed ledger technology presents a means to rive and improve airline scheduling and operations [158]. To this effect, we employ a special type of nonlinear parallel-chain DLT called Hashgraph, which offers four pertinent advantages for achieving adaptive and dynamic decision-making interaction that are not readily available in

current decision support systems for airline operations recovery and disruption management [37].

1. **Performance** - The Hashgraph consensus algorithm can process hundreds of thousands of transactions per second, and thus affords an almost-perfect efficiency in bandwidth usage for a fully connected peer-to-peer network of intelligent agents that represent functional roles in SWA-NOC. This ensures that consensus latency (i.e. real time recovery negotiation amongst intelligent agents) is restricted to seconds, as opposed to minutes observed in existing approaches for simultaneously-integrated recovery in airline disruption management [9].

2. **Security** - Existing platforms for simultaneously-integrated recovery in airline disruption management employ consensus methods that require coordinators, leaders and communication downtime to enable interaction amongst agents in a multi-agent system [153]. As such, these platforms are vulnerable to distributed denial of service (DDoS) [187, 188] attacks that compromise the integrity of the multi-agent system by allowing multiple systems to innudate the bandwidth and resources of a targeted system. In that regard, Hashgraph is immune to DDoS attacks against the consensus algorithm because there are no leaders or coordinators in the data infrastructure. As such, Hashgraph enables the ultimate benchmark for security currently attainable in distributed consensus through asynchronous byzantine fault tolerance (ABFT) [37, 139, 189]. ABFT ensures that the consensus order of transactions (i.e. disruption resolutions) match the actual order in which the transactions posted to the distributed ledger are resolved by intelligent agents. To that effect, it is almost impossible for a single intelligent agent to prevent the writing of transaction information to the distributed ledger, or influence the order of transactions from consequent consensus in the multi-agent system.

3. **Governance** - The decentralized AI network hosted on the Hashgraph platform is governed concurrently by all eleven functional roles in the AOCC that rep-

resent intelligent agents in a multi-agent system, wherein no intelligent agent has control and no small group of intelligent agents has exorbitant influence over the multi-agent system as a whole. This eliminates the need for centralized control observed in many existing monolithic systems for airline disruption management.

4. **Stability** - Hashgraph presents a technology and platform that ensures that intelligent agents automatically validate the ancestry of the information circulated on the distributed ledger prior to deployment through a shared state mechanism. As such, the state mechanism on the Hashgraph platform for airline disruption management is defined by the UTFM and PTFM, which represent unilateral technical controllers in the decentralized AI network during schedule recovery. Furthermore, Hashgraph enables human specialists or supervisors (i.e. platform and software developers) in the AOCC to specify changes and updates to components of intelligent agents, such that the updates are automatically adopted for all intelligent agents at precisely the same time. This ensures that any intelligent agent with antiquated updates is unable to modify the distributed ledger or tender its version of the ledger as valid.

Interaction in SWA-NOC as a Byzantine Generals Problem

Solving the Byzantine Generals Problem [39] represents one of the most challenging methods for verifying and validating the reliability of a computerized multi-agent system in order to manage the failure of one or more of its constituent agents. Conceptually, the Byzantine Generals Problem describes the communication of separate generals in several divisions of the Byzantine army camped outside an enemy city, with only one messenger available to each general. The objective of the generals' mission is to decide on the best collective plan of action after each general observes the behavior (i.e. stratagem) of the enemy. However, some generals may be betrayers that try to hinder loyal generals from achieving a consensus on the best plan of

action to defeat the enemy. Thus, solving the Byzantine Generals Problem is analogous to enabling interaction amongst functional roles in SWA-NOC during disruption management. In this analog, each intelligent agent for a functional role represents a Byzantine general and the enemy represents a disruption in schedule and operations at a particular airline station. The collective plan of action (i.e. recovery plan) describes the order in which disruption resolutions from multiple functional roles (or intelligent agents) are implemented. As such, the messenger for an intelligent agent is defined by cryptographic keys [190] that encode disruption resolutions from appropriate AI models imbued in the intelligent agent. In that regard, the objective of the interaction amongst intelligent agents in a multi-agent system for airline disruption management is to agree on the best recovery plan of action for a disrupted airline schedule such that the following conditions are upheld:

- All intelligent agents representative of separate functional roles in SWA-NOC decide upon the same recovery plan of action.
- A small number of traitors (i.e. intelligent agents) cannot cause loyal intelligent agents to employ a bad recovery plan.

Thus, the intelligent agents operating in a decentralized multi-agent system must have a robust algorithm to guarantee these conditions. To that effect, the Swirlds (Hashgraph) consensus algorithm provides an appropriate medium for insuring that the aforementioned conditions are always satisfied during disruption management and operations recovery.

Hashgraph for Consensus in SWA-NOC

The algorithms that solve the Byzantine agreement problem typically exchange many messages for intelligent agents to vote. For many multi-agent systems, a single YES/NO decision by n intelligent agents can require up to $O(n^3)$ messages to be sent across the network [9, 191]. And extending an algorithm to decide a total

order on a set of transactions for a single YES/NO decision can further increase the voting traffic and compound latency problems. However, the Swirlds consensus algorithm [37], which addresses Byzantine agreement on the Hashgraph DLT, employs a virtual voting mechanism that sends zero votes over the network for a multi-agent system. As such, we define pertinent concepts that enable the applicability of the Swirlds (Hashgraph) consensus algorithm to the decentralized AI network for airline disruption management as follows [45]:

- *Transactions*: A transaction occurs when any intelligent agent representative of a functional role in SWA-NOC publishes its disruption resolution data and corresponding timestamp (i.e. time when the data is appended) to the distributed Hashgraph ledger. The disruption resolution data contains two pieces of information namely:

  1. The recovery uncertainty or reliability measure (i.e. entropy) of the set of actions for disruption resolution generated by the UTFM in the intelligent agent.
  2. The recovery impact measured in terms of turnaround duration, block time duration, tactical delay duration, and strategic delay duration estimated by the PTFM in the intelligent agent.

  Any intelligent agent in the multi-agent system (i.e. decentralized AI network) can create a signed transaction at any time when a disruption in scheduled airline operations occurs. All intelligent agents in the multi-agent system receive a copy of the transaction, and the decentralized AI network reaches Byzantine agreement on the order of those transactions.

- *Fairness*: This ensures that it is difficult for a small group of dubious intelligent agents to unfairly influence the order of transactions that is selected as consensus.

- *Gossip*: This represents how disruption resolution information is disseminated by each intelligent agent repeatedly selecting another intelligent agent at random, and telling them all they know.

- *Hashgraph*: This represents the data structure or ledger upon which transaction records are gossiped and in what order they are gossiped.

- *Gossip-about-gossip*: This indicates the gossip protocol that the *Hashgraph* employs for its operation. The information being gossiped is the history of the gossip itself, and not the disruption resolution data contained in the gossip. As such, only a small amount of bandwidth is required for gossiping transactions amongst intelligent agents.

- *Virtual voting*: Every intelligent agent is privy to a copy of the *Hashgraph*. For instance, an intelligent agent that represents the Security functional role can estimate the vote that another intelligent agent representative of the Ground Operations functional role would have sent if they both had been executing a traditional Byzantine agreement protocol that required them to send votes. As such, the intelligent agent representing Ground Operations does not need the intelligent agent representing the Security functional role to actually vote. To that effect, every intelligent agent can attain Byzantine agreement on any number of recovery plan actions without a single vote ever being sent. The *Hashgraph* data structure alone is sufficient, so no additional bandwidth is expended beyond that required for gossiping the data structure.

- *Famous witnesses*: There are situations during interaction in the multi-agent system where a list of n transactions are ordered by running separate Byzantine agreement protocols on $\mathrm{O}(n \log n)$ different YES/NO questions that seek to answer whether a disruption resolution event x occurred before another disruption resolution event y. To address these situations in an expedient manner, a few disruption resolution events (i.e. vertices in the *Hashgraph*) called *witnesses*

are selected that define a witness to be *famous* if the *Hashgraph* data structure shows that majority of the participating intelligent agents on the ledger received a particular disruption resolution event fairly soon after it was created. Thus, it is sufficient to execute the Byzantine agreement protocol only for witnesses that decide whether or not a particular witness is famous for each intelligent agent. As such, it becomes relatively easy to retrieve a fair total order for all disruption resolution events upon attaining Byzantine agreement on the exact set of famous witnesses.

- *Strongly seeing*: This property is defined as the ability of any two disruption resolution events (i.e. vertices) x and y in the *Hashgraph* to be connected by multiple directed paths passing through enough participating intelligent agents on the distributed ledger. For instance, if the intelligent agents for the Security and Ground Operations functional roles are able to independently estimate the virtual vote on a specific question from the intelligent agent representing the Weather functional role, then the intelligent agents for the Security and Ground Operations functional role will get the same answer. The proof of the lemma that demonstrates this proposition forms the foundation for the subsequent mathematical proof of Byzantine agreement with a probability of one, which is achieved by the Swirlds consensus algorithm [192].

- *Asynchronous Byzantine Fault Tolerance*: The Byzantine fault tolerance theorem states that there is absolute certainty that each disruption resolution event x created by an honest intelligent agent will eventually be assigned a consensus position in the total order of disruption events. Asynchrony for Byzantine fault tolerance is ensured based upon the assumption that the digital signatures and cryptographic hashes are secured, such that signatures can not be forged, signed messages can not be altered without being detected, and hash impingement can never be found [193]. As such, syncing the gossip protocol ensures that when an intelligent agent a sends intelligent agent b all the disruption resolution data it

knows, b accepts only those that have a valid signature and contain valid hashes corresponding to the disruption resolution data that it has available. Therefore, the following definitions describe the property of asynchronous Byzantine fault tolerance in the multi-agent system for obtaining a recovery plan during airline disruption management:

1. A disruption resolution event x is defined to be an ancestor of a disruption resolution event y if x is y, a parent of y, a parent of a parent of y and so on. Furthermore, x is also a self-ancestor of y if x is y, or a self-parent of y, or a self-parent of a self-parent of y and so on.
2. The *round created number* (i.e. round) of a disruption resolution event x is $r + i$, where r is the maximum round number of the parents of x or 1 if it has no parents, and i is 1 if x can strongly see more than $2n/3$ witnesses in round r or 0 otherwise. Note that n is the total number of participating intelligent agents.
3. The *round received number* (i.e. received round) of a disruption resolution event x is the first round where all unique famous witnesses are descendants of x.
4. The pair of disruption resolution events (x, y) is a fork (i.e. a collectively agreed upon intelligent agent update) if x and y have the same creator but neither of them is a self-ancestor of the other.
5. An honest intelligent agent participant on the distributed ledger tries to sync infinitely often with every other participating intelligent agent, creates a valid disruption resolution event after each sync with cryptographic hashes of the latest self-parent and other parents, and will never create two disruption resolution events that are forks with each other.
6. A disruption resolution event y can be seen by a disruption resolution event x if y is an ancestor of x, and the ancestors of x do not include a fork by the creator of y.

7. A disruption resolution event x can *strongly see* disruption resolution event y if x can see y and there is a set of disruption resolution events R from more than two-thirds of participant intelligent agents, such that x can see every disruption resolution event in R and y can be seen by every disruption resolution event in R.
8. A witness is the first disruption resolution event created by an intelligent agent in a round.
9. A unique famous witness is a *famous witness* that does not have the same creator as any other famous witness created in the same round.
10. *Hashgraph* P for intelligent agent a and *Hashgraph* Q for intelligent agent b are said to be consistent (i.e. exactly the same) if and only if for any disruption resolution event x in both *Hashgraphs*, there are the same set of ancestors for x with the same parent and self-parent edges between the set of ancestors.

- *Consensus*: The stake of each intelligent agent is defined by a positive integer that represents the total entropy (i.e. information) of the set of recovery activities generated by its UTFM. As such, the vote of an intelligent agent in the multi-agent system, while creating a recovery plan, is weighted proportional to its voting stake during each round of voting. Thus, consensus in the multi-agent system is defined by a set of intelligent agents whose combined voting stake is more than $2n/3$, where n is the total stake of all participating intelligent agents. In complement, the consensus timestamps of disruption resolution events for a set of disruption resolution events R is the median of the timestamps in R weighted by the voting stake of the creator of each event in R. As such, the weighted median is analogous to selecting each disruption resolution event x in R and placing multiple copies of the timestamp of x into a basket, such that the number of timestamp copies is equal to the stake of the intelligent agent that created y, and then choosing the median of the timestamps in the basket.

Table 5.2. Position weights of states at separate phases of activity in the UTFM for an intelligent agent

ADM Phase	Possible States	Position Weight
Tactical	TAS, TOS, ES, TIS	1
Operational	TAD, TOD, ED, TID	4
Strategic	TAO, TOO, EO, TIO	2

These routines form the basis for proof of stake in achieving consensus in the multi-agent system.

All mathematical proofs, lemmas, and algorithmic routines that enable all the aforementioned concepts for the Hashgraph consensus framework can be found in [45].

Defining Stake for Functional Roles in SWA-NOC

Recall from Chapter 4 that the UTFM is comprised of a total of 12 interconnected states for three separate phases of activity; that is, four schedule states for tactical disruption management, four decision states for operational disruption management, and four outcome states for strategic disruption management. Furthermore, the stake of an intelligent agent operating on the Hashgraph consensus platform must be expressed as a positive integer [45]. As such, we employ a method that measures the stake of a functional role in the AOCC as the total entropy of the trace from the first schedule state (i.e. Turnaround Schedule or TAS) to the last outcome state (i.e. Taxi-Out Outcome or TIO), given a set of input criteria for each phase of activity. Mathematically, the entropy E of a discrete random variable T representing the trace with probability mass function $t(x)$ is expressed as:

$$E(T) = -\sum_{x} t(x) \log_2 t(x) \tag{5.11}$$

Thus, the entropy is the expected number of bits necessary to communicate the value of the trace T if the best possible method (i.e. coding scheme) is used for estimating the distribution of $t(x)$. As such, we estimate $t(x)$ as the probability of the most likely sequence of transitions (i.e. the Viterbi likelihood) from Turnaround Schedule to Taxi-In Outcome for a given set of input (or action) criteria x provided by a human specialist in the AOCC. However, we can not guarantee that $t(x)$ represents the globally optimal distribution for communicating the value of trace T because the best possible "coding scheme" for $t(x)$ was obtained through Baum-Welch training, which only guarantees local optimality. Thus, we create a surrogate value S of trace T based upon domain knowledge that describes the significance of the value management of flight operation at separate phases of activity in airline disruption management. To estimate the surrogate value, we assign a position weight [194, 195] to states at each phase of activity, such that the lowest position weight is assigned to states for tactical disruption management and the highest position weight is assigned to states for operational disruption management. States for tactical and strategic disruption management receive lower position weights, as compared to states for operational disruption management, because these states are only active prior to the execution of a flight schedule and do not rapidly diminish the real-time value of flight recovery. Table 5.2 shows the position weights for states at distinct phases of activity in the UTFM for a functional role in the AOCC. Since entropy is measured in bits, we adopt a binary position weighting scheme for any trace from Turnaround Schedule to Taxi-Out Schedule, such that each state observed for tactical and strategic disruption management is assigned a value of 1 and 2 respectively, while each state observed for operational disruption management is assigned a value of 4. As such, S represents the sum of the position weights $s(x)$ for N transitions from Turnaround Schedule to Taxi-Out Outcome by following the Viterbi path (i.e. the most likely sequence of actions from Turnaround Schedule to Taxi-Out Outcome obtained through Viterbi decoding). To this end, we amend the entropy estimated in Eqn. 5.11 to information cross entropy, which defines the expected number of bits necessary to communicate

the value (or information) taken by T for employing the plausibly sub-optimal Baum-Welch coding scheme defined by $t(x)$. Mathematically, the information cross entropy (ICE) is expressed as:

$$ICE(s,t) = -S\log_2 t(x) = -N\sum_x s(x)\log_2 t(x) \tag{5.12}$$

Thus, the voting stake v_s of a functional role in the AOCC on a single transaction during airline disruption management is expressed as:

$$v_s = \lfloor ICE \rfloor \tag{5.13}$$

5.2.3 Results and Discussion

This section describes the results from a demonstration of our decentralized AI platform for a multi-agent system that creates airline disruption management and schedule recovery plans, through the interaction of intelligent agents that perform transactions on the Hashgraph data structure and consensus platform. In that regard, the following assumptions, conditions and terms apply for our demonstration:

1. We use a data subset that represents 1% of the *disrupted* data set to create a disrupted airline route network, which involves multiple functional roles in SWA-NOC. This data subset represents disrupted flight schedules that executed sometime between September 2016 and September 2017 in the Southwest Airlines route network, which are not used for developing (i.e. calibrating or training) the UTFM and PTFM for intelligent agents.

2. We assume that the disrupted flight schedules define all or part of a "fictitious" and arbitrary route network served by Southwest Airlines in a representative day of the year.

3. A new transaction, created by an intelligent agent representative of a functional role in the AOCC, is stored as random bytes on the *Hashgraph* ledger,

Table 5.3. Total number of queued disrupted flight schedules for each functional role in SWA-NOC used for case study

Functional Domain	**Total Number of Queued Disrupted Flight Schedules**	**Affected Problem Dimension**
Customer Hold	469	Aircraft and Passenger
Dispatch CSC	175	Aircraft and Crew
Flight Operations	364	Crew
Fuel Management	49	Aircraft
Ground Operations	1684	Aircraft and Passenger
Inflight	795	Crew
Maintenance	336	Aircraft
NAS	227	All
Security	30	Passenger
Technology	90	All
Weather	127	All

based upon the queue number of a specific disruption in the irregular operations (IROPS) database of the functional role.

4. We assume that all queued disrupted flight schedules in the database of the functional role for each intelligent agent have a set of corresponding disruption resolutions (i.e. input criteria) provided by human specialists in the AOCC. With respect to the *disrupted* data set, these input criteria represent actual disruption resolutions used to recover the disrupted flight schedules that executed at a particular time, and serve as concurrent inputs for the UTFM and PTFM.

5. We assume that all human specialists in the AOCC, who provide input criteria for the decentralized AI platform (or disruption resolutions for the multi-agent system), are flexible and expert decision-makers in their respective functional

roles. Hence, they are capable of expediently proffering *rules-of-thumb* to address disruptions that affect their respective functional roles.

Table 5.3 shows the total amount of queued disrupted flight schedules for each functional role in the Southwest Airlines network operations control center. Recall that Southwest Airlines operates over $4,000$ flight schedules every day [48]. As such, there are a total of $4,364$ disrupted flight schedules, which represent the worst plausible IROPS scenario where all flight schedules in the arbitrary airline route network are disrupted at the same time on a representative day of the year. We employ the Python programming language for the amalgamation of the Hashgraph consensus algorithm and the AI models for the intelligent agents that represent functional roles in SWA-NOC, which enables the operation and deployment of the decentralized AI platform for airline disruption management.

Data Structure

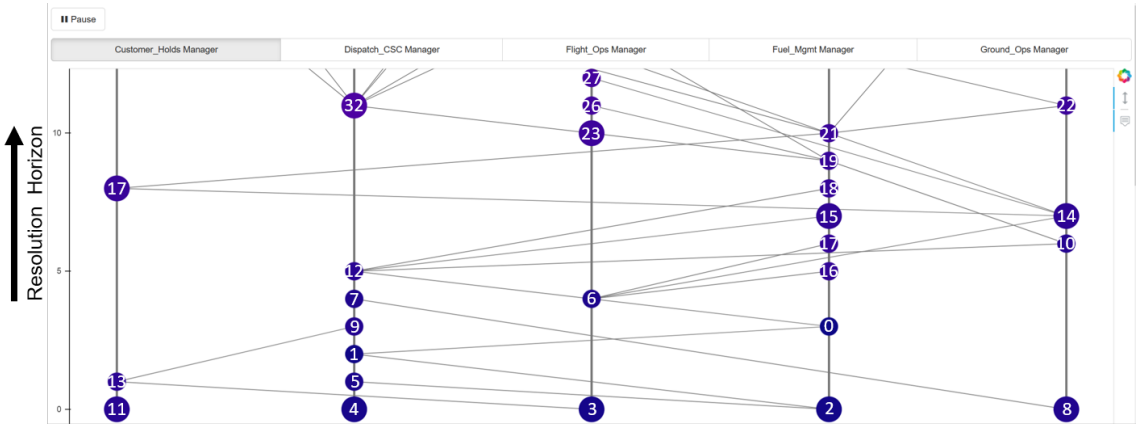

Figure 5.4. Hashgraph data structure revealing a recovery plan from a few functional roles in SWA-NOC

Fig. 5.4 shows the *Hashgraph* (i.e. distributed ledger data structure) for a creating a simultaneously-integrated recovery plan that involves five of the eleven functional roles in SWA-NOC. We use five functional roles for this specific demonstration for

ease of readability. As such, there are five participating intelligent agents (or domain managers) that represent the following functional roles: Customer Hold, Dispatch CSC, Flight Operations, Fuel Management, and Ground Operations, respectively. The circles shown in Fig. 5.4 represent disruption resolution events, such that the larger circles are disruption resolution events that are also *famous witnesses*. In addition, the dark blue circles towards the bottom of the ledger shown in Fig. 5.4 represent disruption resolution events from the first round of consensus, while the purple circles located towards the top of the ledger represent resolution events from the second round of consensus. Furthermore, the non-negative integers shown in Fig. 5.4 represent the position (or place) of a specific disruption resolution event in the consensus order of disruption resolution events, which describe the recovery plan of action during disruption management by the five functional roles. The resolution horizon, which goes from bottom to top (i.e. moving upwards) as shown in Fig. 5.4, describes the manner in which a disruption resolution event or transaction is added to the distributed ledger (i.e. timestamp) by an intelligent agent. Hence, the grey lines represent the gossip paths that indicate how disruption events occur and how corresponding resolution information is disseminated amongst the intelligent domain managers. Therefore, the recovery plan of action agreed upon by the five functional roles, as shown in Fig. 5.4, is to apply the resolution for the second disruption encountered by the Fuel Management functional role first (i.e. small circle labeled 0) before applying the resolution for the third disruption encountered by the Dispatch CSC functional role next (i.e. small circle labeled 1), and so on and so forth. In that vein, the recovery plan of action represents a trace r that follows a set of non-negative integers in ascending order, p, such that each integer in p represents the label of a corresponding disruption resolution event, as shown in Fig. 5.4. Therefore, the recovery plan of action from all five functional roles is expressed as:

$$r = \{0, 1, 2, ..., 32\} = \{\, p \mid p \text{ is non-negative} \,\} \tag{5.14}$$

Table 5.4. Summaries of the first ten disruption resolution events from Hashgraph data structure shown in Fig. 5.4

Consensus Position	**Famous Witness**	**Arbitrary Flight ID**	**Functional Domain**	**Estimated Tactical Delay Duration (mins)**	**Estimated Turnaround Duration (mins)**	**Estimated Block Time Duration (mins)**	**Estimated Strategic Delay Duration (mins)**
0	NO	1536	*Fuel Management*	61	18	126	26
1	NO	3222	*Dispatch CSC*	41	22	93	8
2	YES	2201	*Fuel Management*	7	17	247	33
3	YES	11670	*Flight Operations*	35	1	106	-14
4	YES	12388	*Dispatch CSC*	17	8	294	39
5	NO	15693	*Dispatch CSC*	22	1	106	6
6	NO	34464	*Flight Operations*	55	10	96	6
7	NO	15505	*Dispatch CSC*	0	3	287	40
8	YES	13753	*Ground Operations*	-20	41	339	38
9	NO	5905	*Dispatch CSC*	51	24	103	8

Fig. 5.4 represents a snapshot in a dynamic and continuous timestamp horizon that infinitely increases the set r as new disruptions occur for different functional roles in the AOCC. As such, any number in the trace sequence r that is not visible in Fig. 5.4 is further up along the resolution horizon. Note that the circles shown in Fig. 5.4 are unlabeled and all have the same size prior to achieving consensus on the Hashgraph platform.

Table 5.4 reveals the summaries of the first ten disruption resolution events (i.e. transactions) in the consensus recovery plan shown in Fig. 5.4. Of the ten disruption resolution events, four of them (i.e. $2, 3, 4, 8$) are famous witnesses. Thus, the first (parent) disruption resolution events registered by intelligent domain managers on the Hashgraph ledger are for disrupted flight schedules with the following identification numbers (i.e. flight id) respectively: $2201, 11670, 12388$, and 13753. The highest tactical delay (i.e. delay applied before aircraft boarding and departure) of $61mins$ was estimated for the recovery (or management) of disrupted flight 1536 by the Fuel Management functional role. Conversely, the least tactical delay estimated among the ten disrupted flights from Table 5.4 represents the execution of flight 13753 twenty minutes earlier than originally planned by the Ground Operations functional role in the AOCC. Congruently, the highest strategic delay (i.e. delay applied after aircraft arrival and deplaning) of $40mins$ was estimated for the recovery of disrupted flight 15505 by the Dispatch CSC domain manager. Furthermore, the lowest strategic delay estimated among the ten disrupted flight schedules, highlighted in Table 5.4, affirms the arrival of flight 11670 fourteen minutes earlier than the originally scheduled arrival time prior to disruption, as estimated by the Flight Operations functional domain manager. The total tactical and strategic delay estimated for recovering the ten flights are $269mins$ and $190mins$ respectively, as evidenced in Table 5.4. As such, the recovery plan (from the decentralized AI platform) for the first ten flights will cost the airline an average of \$359.55 per disrupted passenger, assuming $\$47/hr$ [196] as the average value of a passenger's time on each disrupted flight and the passenger flew on all ten disrupted flights.

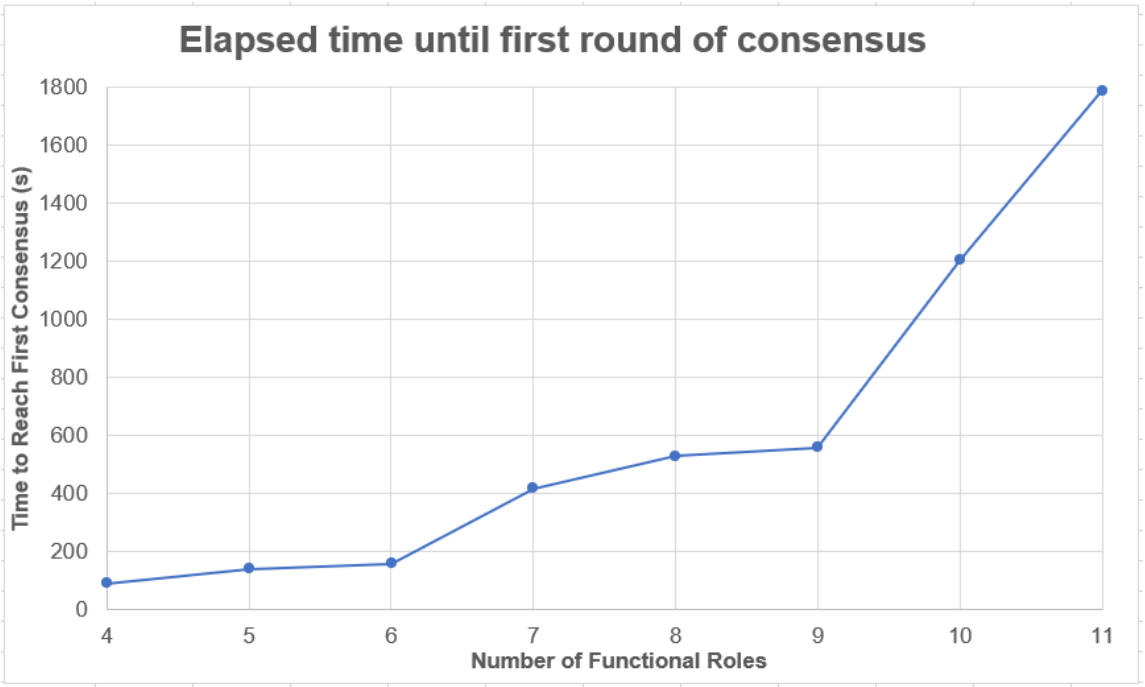

Figure 5.5. Elapsed time period before first round of consensus for an increasing number of functional roles in SWA-NOC

Performance

Fig. 5.5 shows a plot of the period of time that elapsed before the first round of consensus is attained and registered on the Hashgraph platform, as the number of functional roles involved in disruption management (or operations recovery) is increased from four to eleven. Note that the number of functional roles, shown on the x-axis in Fig. 5.5, is increased based upon the alphabetical order of all the functional roles in SWA-NOC. As such, the scenario with four functional roles indicates an interaction of the following functional roles on the Hashgraph platform: Customer Hold, Dispatch CSC, Flight Operations, and Fuel Management. In that vein, the scenario with five functional roles indicates an interaction of the Customer Hold, Dispatch CSC, Flight Operations, Fuel Management, and Ground Operations functional roles on the Hashgraph consensus platform. Since the Hashgraph consensus timeline continues infinitely, we measure the time to reach first consensus in lieu. Hence, the

time to reach first consensus, shown on the y-axis in Fig. 5.5, represents the duration from the start of the interaction amongst intelligent domain managers until the time when all famous witnesses in the first round are assigned a consensus position and corresponding timestamp, assuming the throughput that Hashgraph provides remains constant for all scenarios. Fig. 5.5 reveals that the rate at which the elapsed time to reach first consensus increases for interactions amongst nine to eleven functional roles is six times the rate at which the elapsed time to reach first consensus increases for interactions amongst four to six functional roles in the AOCC. As such, the minimum elapsed time to reach first consensus is $90s$ during interaction for simultaneously-integrated recovery amongst four functional roles, and the maximum elapsed time to reach first consensus is $1780s$ during interaction for simultaneously-integrated recovery amongst all eleven functional roles in SWA-NOC.

Chapter Summary: This chapter provided an extensive review of fusion techniques for artificial intelligence and distributed ledger technology, and how they enabled the creation of a decentralized AI platform that simulates the integration and interaction of intelligent agents for simultaneously-integrated recovery during airline disruption management. Through a synthesis of AI models (i.e. UTFM and PTFM discussed in Chapter 4) and a nonlinear parallel-chain distributed ledger technology called Hashgraph, we developed and assessed the framework of an intelligent multi-agent system that can provide real-time schedule recovery plans based upon *rules-of-thumb* provided by human experts in the airline operations control center.

6. OUTRO

The airline scheduling and disruption management process for distinctive flight operations is an innately iterative profit-maximizing process that occurs over a six to nine month period, and can be separated into tactical planning, operational planning, and strategic planning respectively. Tactical and strategic planning are short-term and long-term scheduling initiatives, respectively, employed by airlines to facilitate the initiatives developed in the operational planning phase (i.e. during execution of planned airline schedule). Many a time, the planned (optimal) flight schedule prior to execution is subject to random disruptive events during execution that invariably complicate schedule recovery typically enabled by large-scale monolithic optimization techniques. Furthermore, the systems upon which these optimization techniques are imbued are centralized and rapidly become inefficient as more complexity (i.e. functionality) is added to the system. As such, these systems are only capable of recovering one or two problem dimensions (i.e. aircraft, crew, passenger) in airline disruption management at a time in a sequential manner. To this end, this dissertation introduced and demonstrated a data-driven and modular framework that engages a unique and systematic design paradigm for enabling simultaneously-integrated recovery of all problem dimensions in airline disruption management, via a multi-agent system modeling of the airline operations control center. Thus, to conclude, this chapter outlines a summary of the research contributions of this dissertation to the field of airline scheduling and disruption management and recommendations of directions for further research studies.

6.1 Summary

The executive synopsis for each chapter in the body of this dissertation are as follows:

6.1.1 A Novel Paradigm for ADM

Contribution

1. We introduced an intelligent multi-agent system (*i-MAS*) framework for airline disruption management that strictly applies historical data on airline scheduling and recovery operations to find optimum flight schedule recovery plans, which are attained through simultaneous interaction of functional domains (i.e. IROPS managers) in the AOCC.

2. We provided an insight of the existing framework for simultaneously-integrated recovery and our (*i-MAS*) framework, to establish the relevance of generic research assignments necessary for enabling a purely data-driven design paradigm for airline disruption management.

3. To the best of our knowledge, our *i-MAS* framework is the first architecture for airline disruption management that employs concurrent virtual voting amongst functional roles in the AOCC – which are affected by separate problem dimensions – to achieve an automatic (or semi-automatic) system for simultaneously-integrated recovery.

Conclusion

Simultaneously-integrated recovery in airline disruption management is a fledgling research area that can benefit from more investigation. Through new design principles and methodologies that leverage the quotidian existence of data and relevant domain information on several aspects of the air transportation system, robust strategies that

address every fundamental tenet of airline disruption management can be readily created and implemented.

6.1.2 Exploratory Data Analysis for a *SIR* Paradigm in ADM

Contribution

1. We introduced and explored several abstraction methods for applying, enhancing, and sequestering raw features and labels in a historical data set on airline scheduling and operations recovery from a major U.S. airline, to readily identify relevant cognitive patterns for key drivers during airline disruption management.

2. We investigated the application of appropriate machine learning techniques for revealing patterns, pertinence, and properties of abstracted data features, which provide necessary a priori information for Bayesian and pseudo-Bayesian methods. These methods are subsequently used for developing functional models in an intelligent multi-agent system for airline disruption management.

Conclusion

Machine learning techniques such as feature transformation and dimensionality reduction provide excellent platforms for verifying empirical processes and validating domain knowledge for airline disruption management. The patterns and information gleaned from the results and observations from visual techniques (such as PCA and t-SNE) identified the viability of seasonal (temporal) and geographical features in the data set as appropriate predictors for AI models; while the results obtained from qualitative methods (such as MIR and GPR) established the importance of these predictors and the underlying processes that define their combination for high fidelity AI models. As such, the findings from Chapter 3 partially address our first research question; through the completion of research tasks that identify the credibility of

exercising machine learning to characterize functional parts of the AOCC as intelligent decision support systems.

6.1.3 Creating Intelligent Agents for a *SIR* Paradigm in ADM

Contribution

1. We adeptly used experience (i.e. historical data on airline schedule and operations recovery) to construct an internal model of the transitions and immediate outcomes of scheduling activities and decisions for different phases of flight operations, by effectively describing the model environment as a relational dynamic Bayesian network (RDBN) architecture. Our architecture defines the interaction between schedule changes and decision-making during airline disruption management, for a unique intelligent agent (i.e. domain manager) in a multi-agent system.

2. We provided a modular approach for implementing an uncertainty transfer function model (UTFM) for disruption management. Our approach inculcates feature engineering and probabilistic graphical modeling methods that enable the use of appropriate machine learning algorithms to effectively calibrate parameters for a RDBN architecture.

3. We expertly used historical data on airline schedule and operations recovery to develop a system of artificial neural networks (ANNs), which define a predictive transfer function model (PTFM) for estimating the recovery impact of disruption resolutions at separate phases of schedule execution during disruption management.

4. We provided a modular approach for assessing and executing a PTFM for airline disruption management, by employing a parallel ensemble method to develop generative routines that amalgamate a system of ANNs.

Conclusion

Predictive analytics provided a set of learning and inference techniques that ensure the accessibility, efficacy and reliability of distinct patterns and behaviors for proactive (tactical) disruption management prior to schedule execution, reactive (operational) disruption management during schedule execution and proactive (strategic) disruption management after schedule execution; all of which are necessary for achieving robust airline disruption management. First, we attained an effective application of two different classes of dynamic programming algorithms, i.e. the Baum-Welch and Viterbi algorithms, to respectively learn and decode the parameters of different HMMs that constitute an overarching relational dynamic Bayesian network called the UTFM. Next, we achieved a suitable implementation of bootstrap aggregation (i.e. bagging) for combining multiple independently trained ANNs that define the PTFM, such that current industry standards for tardiness during flight schedule execution are satisfied. Thus, the findings from Chapter 4 completely address our first and third research questions, respectively, by:

1. Verifying the information retrieved through exploratory data analysis in Chapter 3 and validating the use of the statistics of predictive analytics to evaluate functional parts of the AOCC.

2. Recognizing and assessing consistent metrics, such as delay periods, to measure the performance and effectiveness of disruption resolutions.

6.1.4 Enabling Integration and Interaction for a *SIR* Paradigm in ADM

Contribution

1. We enabled the fusion of artificial intelligence (AI) and distributed ledger technology (DLT) for airline disruption management, by developing the *i-MAS* architecture for multiple functional roles in the AOCC as a decentralized AI platform.

2. We created a protocol to invoke the Hashgraph consensus algorithm, which employs the reliability of any disruption resolution provided by a human specialist as the stake (i.e. relative interest) of the associated functional role during its interaction with other participating functional roles in the AOCC.

3. We established the efficacy of a decentralized AI platform for simultaneously-integrated recovery in airline disruption management, by assessing a scenario of randomly disrupted flight schedules across multiple functional roles and problem dimensions in the AOCC.

Conclusion

Trust in human-AI collaboration for robust airline disruption management is contingent upon expedient verification and validation of *rules-of-thumb* – provided by human specialists in the AOCC – that define input criteria for a decentralized AI platform. Thus, we adopted a synthesis of artificial intelligence and distributed ledger technology to create a symbiotic relationship for the integration and interaction of intelligent agents in the *i-MAS* framework for the AOCC. As such, we achieved a semi-automatic platform that effectively eliminates the drawback of centralization for integration of intelligent agents in the AOCC, by engaging the decentralized and immutable nature of distributed ledger technology to realize simultaneous interaction amongst intelligent agents during airline disruption management. We used the resultant decentralized AI platform to effect simultaneously-integrated recovery for an artificial network of disrupted real-world flight schedules across multiple functional roles in the AOCC, and obtained credible recovery plans within minutes of invoking the *i-MAS* framework. To this effect, the findings from Chapter 5 completely addresses our second research question, by revealing simultaneous decision-making among intelligent functional roles in the AOCC through latest advancements in consensus mechanism design.

6.2 Recommendations for Future Work

Based upon the scope of work presented in this dissertation, we propose the following recommendations for further research in simultaneously-integrated recovery for airline disruption management:

- First, the data used to inform the development of the AI models (i.e. UTFM and PTFM), was provided by one airline that primarily operates a point-to-point route network structure. As such, there is a need to investigate an equivalent framework for creating AI models that can readily appropriate data from any major airline, which utilizes a hub and spoke route network or a point-to-point route network. Moreover, to facilitate system-wide disruption management measures like the FAA collaborative decision making (CDM) initiative, readily accessible data from other air transportation system stakeholders (such as airports) can be inculcated to improve the effectiveness of the existing *i-MAS* framework.

- Second, the selection of specific data features for different phases of activity in the construction of the UTFM is primarily informed by literature and expert inputs of human specialists from one airline, and may contain biases with respect to separate perspectives for different objectives of air transportation stakeholders for system-wide disruption management. As such, proven non-parametric and unsupervised machine learning techniques can be employed to mitigate and validate biases for ensuring a fairly objective selection of features to represent different air transportation system stakeholders for robust disruption management in the national airspace system. Furthermore, the Baum-Welch algorithm presents an inherently sub-optimal unsupervised learning routine for obtaining component HMMs of the UTFM. To that effect, more research to ensure and enhance solution fidelity of unsupervised machine learning methods is most opportune.

REFERENCES

REFERENCES

[1] Henrique Sousa, Ricardo Teixeira, Henrique Lopes Cardoso, and Eugenio Oliveira. Airline Disruption Management. pages 398–405, 2016.

[2] Cynthia Barnhart, Peter Belobaba, and Amedeo Odoni. Applications of Operations Research in the Air Transport Industry. *Transportation Science*, 37(4):368–391, 2003.

[3] Tobias Grosche. *Computational Intelligence in Integrated Airline Scheduling*. 2009.

[4] Niklas Kohl, Allan Larsen, Jesper Larsen, Alex Ross, and Sergey Tiourine. Airline disruption management-Perspectives, experiences and outlook. *Journal of Air Transport Management*, 13(3):149–162, 2007.

[5] António Jesus Monteiro De Castro. Designing a Multi-Agent System for Monitoring and Operations Recovery for Airline Operations Control Centre. *The Lancet*, 367(9512):723, 2006.

[6] Tobias Grosche. Integrated airline scheduling. *Studies in Computational Intelligence*, 173:59–171, 2009.

[7] Nadia Galaske and Reiner Anderl. Disruption Management for Resilient Processes in Cyber-physical Production Systems. *Procedia CIRP*, 50:442–447, 2016.

[8] Jens Clausen, Allan Larsen, Jesper Larsen, and Natalia J. Rezanova. Disruption management in the airline industry-Concepts, models and methods. *Computers and Operations Research*, 37(5):809–821, 2010.

[9] Antonio J M Castro, Ana Paula, Rocha Eugenio, and Eugenio Oliveira. *Studies in Computational Intelligence 562 Ana Paula Rocha A New Approach for Disruption Management in Airline Operations Control*. 2014.

[10] T. Andersson and P. Värbrand. The flight perturbation problem. *Transportation Planning and Technology*, 27(2):91–117, 2004.

[11] Francois Vanderbeck. On Dantzig-Wolfe Decomposition in Integer Programming and ways to Perform Branching in a Branch-and-Price Algorithm. *Operations Research*, 48(1):111–128, 2000.

[12] Zhao Xiuli and Guo Yanchi. Study on GRAPS-ACO algorithm for irregular flight rescheduling. In *Proceedings - 2012 International Conference on Computer Science and Service System, CSSS 2012*, pages 266–269, 2012.

[13] Thomas A. Feo and Mauricio G C Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2):67–71, 1989.

[14] Luca M Gambardella and Marco Dorigo. Ant-Q : A Reinforcement Learning approach to the traveling salesman problem. *Update*, 5625(1):252–260, 1995.

[15] Claude P. Medard and Nidhi Sawhney. Airline crew scheduling from planning to operations. *European Journal of Operational Research*, 183(3):1013–1027, 2007.

[16] Ahmed Abdelghany, Goutham Ekollu, Ram Narasimhan, and Khaled Abdelghany. A proactive crew recovery decision support tool for commercial airlines during irregular operations. *Annals of Operations Research*, 127(1-4):309–331, 2004.

[17] Chunhua Gao, Ellis Johnson, and Barry Smith. Integrated Airline Fleet and Crew Robust Planning. *Transportation Science*, 43(1):2–16, 2009.

[18] A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, 1972.

[19] Stephane Bratu and Cynthia Barnhart. Flight operations recovery: New approaches considering passenger recovery. *Journal of Scheduling*, 9(3):279–298, 2006.

[20] Jon D. Petersen, Gustaf Sölveling, John-Paul Clarke, Ellis L. Johnson, and Sergey Shebalov. An Optimization Approach to Airline Integrated Recovery. *Transportation Science*, 46(4):482–500, 2012.

[21] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[22] Shan Lan, John-Paul Clarke, and Cynthia Barnhart. Planning for Robust Airline Operations: Optimizing Aircraft Routings and Flight Departure Times to Minimize Passenger Disruptions. *Transportation Science*, 40(1):15–28, 2006.

[23] Cynthia Barnhart. Irregular Operations: Schedule Recovery and Robustness. In *The Global Airline Industry*, pages 253–274. 2009.

[24] David Nathans. Efficient operations: Building an operations center from the ground up. In *Designing and Building Security Operations Center*, chapter 1, pages 1–24. Elsevier, 2015.

[25] Norman Rose. Passengers first: Re-thinking irregular operations. page 25, 2014.

[26] Ira Gershkoff. Shaping the future of Airline Disruption Management (IROPS). pages 1–32, 2016.

[27] Mark W Maier. Architecting Principles for Systems-of-Systems. *Systems Engineering*, 1(4):267–284, 1998.

[28] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 2005.

[29] Reza Olfati-Saber, J. Alex Fax, and Richard M. Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[30] C. E. Rasmussen and C K I Williams. *Gaussian Processes for Machine Learning*. 2006.

[31] Christopher M Bishop. *Pattern Recognition and Machine Learning*, volume 4. 2006.

[32] Andy Extance. the Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance. *Nature*, 552(7685):301–302, 2017.

[33] Michael Chui, James Manyika, Mehdi Miremadi, Nicolaus Henke, Rita Chung, Pieter Nel, and Sankalp Malhotra. Notes from the AI frontier. Insights from hundreds of use cases. page 36, 2018.

[34] SITA. Research Into the Usability and Practicalities of Blockchain Technology for the Air Transport Industry. Technical report, 2017.

[35] S. Y. Huh, K. H. Moon, and H. Lee. Data abstraction approach for query relaxation. *Information and Software Technology*, 42(6):407–418, 2000.

[36] Harish Natarajan, Solvej Krause, and Helen Gradstein. *Distributed Ledger Technology and Blockchain*. 2017.

[37] Leemon Baird, Mance Harmon, and Paul Madsen. Hedera: A governing council and public hashgraph network - The trust layer of the internet. *Whitepaper*, pages 1–27, 2018.

[38] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. Making Smart Contracts Smarter. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*, pages 254–269, 2016.

[39] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[40] Steven F. Railsback and Volker Grimm. *Agent-based and individual-based modeling: A practical introduction*. 2011.

[41] Alison J J Heppenstall, Andrew T. Crooks, Linda M. See, and Michael Batty. *Agent-based models of geographical systems*. 2012.

[42] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep Reinforcement Learning for Multi-Agent Systems: A Review of Challenges, Solutions and Applications. (1992):1–27, 2018.

[43] Peter Dayan and Yael Niv. Reinforcement learning: The Good, The Bad and The Ugly, 2008.

[44] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: The Condor experience, 2005.

[45] Leemon Baird. The swirlds hashgraph consensus algorithm: fair, fast, byzantine fault tolerance. *Swirlds Tech Report*, 01:1–28, 2018.

[46] The An Ngo and Linda See. Calibration and validation of agent-based models of land cover change. In *Agent-Based Models of Geographical Systems*. 2012.

[47] Amadeus IT Group. Airline Disruption Management. 2016.

[48] John Hagel, John Seely Brown, Andrew De Maar, and Maggie Wooll. Southwest Airlines: Baker workgroup - Reducing disruption and delay to accelerate performance. 2017.

[49] Raghu Ramakrishnan and Jeffrey D. Ullman. A survey of deductive database systems. *The Journal of Logic Programming*, 23(2):125–149, 1995.

[50] Russell J. Abbott. Knowledge abstraction. *Communications of the ACM*, 30(8):664–671, 1987.

[51] Barbara Liskov. Data Abstraction and Hierarchy. *ACM SIGPLAN Notices*, 23(5):17–34, 1988.

[52] Dennis F.X. Mathaisel. Decision support for airline schedule planning. *Journal of Combinatorial Optimization*, 1(3):251–275, 1997.

[53] H. Liu and H. Motoda. Feature transformation and subset selection. *IEEE expert*, 13(2):26–28, 1998.

[54] Andrew Kusiak. Feature transformation methods in data mining. *IEEE Transactions on Electronics Packaging Manufacturing*, 24(3):214–221, 2001.

[55] G. Shurkhovetskyy, N. Andrienko, G. Andrienko, and G. Fuchs. Data abstraction for visualizing large time series. *Computer Graphics Forum*, 37(1):125–144, 2018.

[56] T. Vincenty. Direct and Inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review*, 1975.

[57] Cedric Seger. An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing. *Degree Project Technology*, page 41, 2018.

[58] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011.

[59] Sanford Weisberg. Yeo-Johnson Power Transformations. *Department of Applied Statistics, University of Minnesota*, 2001.

[60] Cees A.W. Glas. Maximum-likelihood estimation. In *Handbook of Item Response Theory: Volume Two: Statistical Tools*. 2017.

[61] Michel Verleysen and Damien François. The Curse of Dimensionality in Data Mining. *Analysis*, 2005.

[62] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 2008.

[63] L J P Van Der Maaten, E O Postma, and H J Van Den Herik. Dimensionality Reduction: A Comparative Review. *Journal of Machine Learning Research*, 10:1–41, 2009.

[64] Jonathon Shlens. A Tutorial on Principal Component Analysis. 2014.

[65] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 1987.

[66] Laurens Van Der Maaten. Accelerating t-SNE using Tree-Based Algorithms. *Journal of Machine Learning Research*, 15:3221–3245, 2014.

[67] Fernando Pérez-Cruz. Kullback-leibler divergence estimation of continuous distributions. In *IEEE International Symposium on Information Theory - Proceedings*, 2008.

[68] James M. Joyce. Kullback-Leibler Divergence. In *International Encyclopedia of Statistical Science*. 2011.

[69] D Koller and M Sahami. Toward Optimal Feature Selection. In *International Conference on Machine Learning*, 1996.

[70] Michal Moran and Goren Gordon. Curious Feature Selection. *Information Sciences*, 2019.

[71] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson Correlation Coefficient. 2009.

[72] Ai Hua Jiang, Xiu Chang Huang, Zhen Hua Zhang, Jun Li, Zhi Yi Zhang, and Hong Xin Hua. Mutual information algorithms. *Mechanical Systems and Signal Processing*, 24(8):2947–2960, 2010.

[73] Brian C. Ross. Mutual information between discrete and continuous data sets. *PLoS ONE*, 2014.

[74] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, 2004.

[75] Shuyang Gao, Greg Ver Steeg, and Aram Galstyan. Estimating mutual information by local Gaussian approximation. *Uncertainty in Artificial Intelligence - Proceedings of the 31st Conference, UAI 2015*, pages 278–285, 2015.

[76] SheffieldML. GPy Documentation. 2014.

[77] Galit Shmueli and Otto R. Koppius. Predictive analytics in information systems research. *MIS Quarterly: Management Information Systems*, 35(3):553–572, 2011.

[78] Wayne Eckerson. Predictive Analytics: Extending the Value of Your Data Warehousing Investment. *Advances*, 2007.

[79] Todd G. Nick. Descriptive statistics., 2007.

[80] Monica Franzese and Antonella Iuliano. Descriptive statistics. In *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*. 2018.

[81] Louise Francis. Unsupervised learning. In *Predictive Modeling Applications in Actuarial Science: Volume I: Predictive Modeling Techniques*. 2014.

[82] Andreas Glöckner, Benjamin E. Hilbig, and Marc Jekel. What is adaptive about adaptive decision making? A parallel constraint satisfaction account. *Cognition*, 2014.

[83] Paul C. Clements and David M. Weiss. The Modular Structure of Complex Systems. *IEEE Transactions on Software Engineering*, 1985.

[84] Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2007.

[85] Craig R Fox and Gulden Ulkumen. Distinguishing two dimensions of uncertainty. *Perspectives on Thinking, Judging, and Decision Making*, 2011.

[86] Daphne Koller and Nir Friedman. Probabilistic Graphical Models: Principles and Techniques. 2009.

[87] Jc Pomerol. Artificial intelligence and human decision making. *European Journal of Operational Research*, 2217(96):1–28, 1997.

[88] Morten Frydenberg. The chain graph Markov property. *Scandinavian journal of statistics*, 1990.

[89] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *IJCAI International Joint Conference on Artificial Intelligence*, 1999.

[90] Sumit Sanghai, Pedro Domingos, and Daniel Weld. Relational dynamic bayesian networks. *Journal of Artificial Intelligence Research*, 24:759–797, 2005.

[91] Jennifer Neville and David Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 2007.

[92] Wai Ki Ching, Michael K. Ng, and Eric S. Fung. Higher-order multivariate Markov chains and their applications. *Linear Algebra and Its Applications*, 428(2-3):492–507, 2008.

[93] Fei Cao, Barrett R. Bryant, Carol C. Burt, Rajeev R. Raje, Andrew M. Olson, and Mikhail Auguston. A component assembly approach based on aspect-oriented generative domain modeling. *Electronic Notes in Theoretical Computer Science*, 114(SPEC. ISS.):119–136, 2005.

[94] Scott C. Chase. Generative design tools for novice designers: Issues for selection. In *Automation in Construction*, 2005.

[95] Krzysztof Czarnecki. Overview of generative software development. In *Lecture Notes in Computer Science*, 2005.

[96] C. Reid Turner, Alfonso Fuggetta, Luigi Lavazza, and Alexander L. Wolf. A conceptual basis for feature engineering. *Journal of Systems and Software*, 49(1):3–15, 1999.

[97] Enrique Vidal, Frank Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. Probabilistic finite-state machines - Part II, 2005.

[98] Zoubin Ghahramani. An Introduction to Hidden Markov Models and Bayesian Netweoks. *International Journal of Pattern Recognition and Artificial Intelligence*, 2001.

[99] Ben Letham and Cynthia Rudin. Probabilistic Modeling and Bayesian Analysis. *Prediction: Machine Learning and Statistics Lecture Notes*, pages 1–42, 2012.

[100] Jie Yang, Yangsheng Xu, and Chiou S. Chen. Human action learning via hidden Markov model. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.*, 27(1):34–44, 1997.

[101] Leonard E. Baum and Ted Petrie. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, 2007.

[102] JeffA. Bilmes. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. 42(5):42–45, 2011.

[103] Yves Boussemart, Jonathan Las Fargeas, Mary L. Cummings, and Nicholas Roy. Comparing Learning Techniques for Hidden Markov Models of Human Supervisory Control Behavior. 2012.

[104] Andrew J. Viterbi. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, 1967.

[105] G. David Forney. The Viterbi Algorithm. *Proceedings of the IEEE*, 1973.

[106] Jim K. Omura. On the Viterbi Decoding Algorithm. *IEEE Transactions on Information Theory*, 1969.

[107] Olle Haggstrom. *Finite Markov Chains and Algorithmic Applications*. 2002.

[108] Jacob Schreiber. pomegranate Documentation. 2016.

[109] Michael Dudley Delano Clarke. Irregular airline operations: a review of the state-of-the-practice in airline operations control centers. *Journal of Air Transport Management*, 1998.

[110] Alan H Midkiff, R John Hansman, and Tom G Reynolds. Air Carrier Flight Operations. Technical Report July, MIT International Center for Air Transportation, Cambridge, MA, 2004.

[111] Lu Hao and Mark Hansen. How airlines set scheduled block times. In *Proceedings of the 10th USA/Europe Air Traffic Management Research and Development Seminar, ATM 2013*, 2013.

[112] R Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*, 2(0):1137–1143, 1995.

[113] Kolawole E. Ogunsina, Marios Papamichalis, Ilias Bilionis, and Daniel A. DeLaurentis. Hidden Markov Models for Pattern Learning and Recognition in a Data-Driven Model for Airline Disruption Management. 2019.

[114] Lavanya Marla, Bo Vaaben, and Cynthia Barnhart. Integrated disruption management and flight planning to trade off delays and fuel burn. *Transportation Science*, 51(1):88–111, 2017.

[115] Jane Lee, Lavanya Marla, and Alexandre Jacquillat. Dynamic Airline Disruption Management Under Airport Operating Uncertainty. *SSRN Electronic Journal*, 2007:1–41, 2018.

[116] Roger Beatty, Rose Hsu, Lee Berry, and James Rome. Preliminary Evaluation of Flight Delay Propagation through an Airline Schedule. *Air Traffic Control Quarterly*, 1999.

[117] Yu Zhang and Mark Hansen. Real-time intermodal substitution: Strategy for airline recovery from schedule perturbation and for mitigation of airport congestion. *Transportation Research Record*, 2008.

[118] Hartmut Fricke and Michael Schultz. Delay impacts onto turnaround performance. *Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM2009)*, 2009.

[119] Poornima Balakrishna, Rajesh Ganesan, and Lance Sherry. Accuracy of reinforcement learning algorithms for predicting aircraft taxi-out times: A case-study of Tampa Bay departures. *Transportation Research Part C: Emerging Technologies*, 2010.

[120] Bojia Ye, Bo Liu, Yong Tian, and Lili Wan. A methodology for predicting aggregate flight departure delays in airports based on supervised learning. *Sustainability (Switzerland)*, 2020.

[121] João P.S. Rosa, Daniel J.D. Guerra, Nuno C.G. Horta, Ricardo M.F. Martins, and Nuno C.C. Lourenço. *Overview of Artificial Neural Networks*. 2020.

[122] Jinming Zou, Yi Han, and Sung Sau So. Overview of artificial neural networks, 2008.

[123] E W Weisstein. Sigmoid Function. *MathWorld - A Wolfram Web resource*, 2006.

[124] Earl Hunt, Marvin Minsky, and Seymour Papert. Perceptrons. *The American Journal of Psychology*, 1971.

[125] Bernard Widrow and Michael A. Lehr. 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation. *Proceedings of the IEEE*, 1990.

[126] Radford M. Neal. Priors for Infinite Networks. 1996.

[127] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as Gaussian processes. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

[128] Alex M. Andrew. Backpropagation. *Kybernetes*, 2001.

[129] Jeff Heaton. *Artificial Intelligence for Humans, Volume 3: Neural Networks and Deep Learning*. 2015.

[130] Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 1964.

[131] Shie Mannor, Bori Peleg, and Reuven Rubinstein. The cross entropy method for classification. In *ICML 2005 - Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[132] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[133] Shashi Shekhar and Hui Xiong. Root-Mean-Square Error. In *Encyclopedia of GIS*. 2008.

[134] Peter A. Flach and Meelis Kull. Precision-Recall-Gain curves: PR analysis done right. In *Advances in Neural Information Processing Systems*, 2015.

[135] Zhi Hua Zhou. *Ensemble methods: Foundations and algorithms*. 2012.

[136] Nikhil Ketkar and Nikhil Ketkar. Introduction to PyTorch. In *Deep Learning with Python*. 2017.

[137] PyTorch Community. Tensors and Dynamic neural networks in Python with strong GPU acceleration. *Github*, 2016.

[138] Kathleen L. Mosier and Linda J. Skitka. Human decision makers and automated decision aids: Made for each other? In *Automation and Human Performance: Theory and Applications*. 2018.

[139] Miguel Castro and Barbara Liskov. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Transactions on Computer Systems*, 20(4):398–461, 2002.

[140] Melanie Swan. Blockchain Thinking : the Brain as a Decentralized Autonomous Corporation [Commentary], 2015.

[141] Tsan Ming Choi, Stein W. Wallace, and Yulan Wang. Big Data Analytics in Operations Management. *Production and Operations Management*, 27(10):1868–1883, 2018.

[142] Amy Maxmen. AI researchers embrace Bitcoin technology to share medical data, 2018.

[143] Ercan Oztemel and Samet Gursev. Literature review of Industry 4.0 and related technologies, 2020.

[144] E. T. Jaynes. Probability Theory: The Logic of Science. *The Mathematical Intelligencer*, 27(2):83–83, 2003.

[145] Michela Piccarozzi, Barbara Aquilani, and Corrado Gatti. Industry 4.0 in management studies: A systematic literature review. *Sustainability (Switzerland)*, 2018.

[146] Yulin Liu, Yi Liu, Mark Hansen, Alexey Pozdnukhov, and Danqing Zhang. Using machine learning to analyze air traffic management actions: Ground delay program case study. *Transportation Research Part E: Logistics and Transportation Review*, 2019.

[147] Douwe Kiela, Luana Bulat, Anita L. Vero, and Stephen Clark. Virtual Embodiment: A Scalable Long-Term Strategy for Artificial Intelligence Research. (Nips), 2016.

[148] Li Li, Yi Lun Lin, Nan Ning Zheng, Fei Yue Wang, Yuehu Liu, Dongpu Cao, Kunfeng Wang, and Wu Ling Huang. Artificial intelligence test: a case study of intelligent vehicles. *Artificial Intelligence Review*, 2018.

[149] Michel Rauchs, Andrew Glidden, Brian Gordon, Gina C. Pieters, Martino Recanatini, François Rostand, Kathryn Vagneur, and Bryan Zheng Zhang. Distributed Ledger Technology Systems: A Conceptual Framework. *SSRN Electronic Journal*, (August), 2018.

[150] Roger Maull, Phil Godsiff, Catherine Mulligan, Alan Brown, and Beth Kewell. Distributed ledger technology: Applications and implications. *Strategic Change*, 26(5):481–489, 2017.

[151] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System — Satoshi Nakamoto Institute. Technical report, 2008.

[152] Craig S Wright. Bitcoin: A Peer-to-Peer Electronic Cash System. *SSRN Electronic Journal*, 2019.

[153] Soufiane Bouarfa, Jasper Müller, and Henk Blom. Evaluation of a Multi-Agent System approach to airline disruption management. *Journal of Air Transport Management*, 2018.

[154] Stefan Janson, Daniel Merkle, and Martin Middendorf. A decentralization approach for swarm intelligence algorithms in networks applied to multi swarm PSO. *International Journal of Intelligent Computing and Cybernetics*, 2008.

[155] Michael Ball, Cynthia Barnhart, George Nemhauser, and Amedeo Odoni. Air Transportation : Irregular Operations and Control. *Handbooks of Operations Research and Management*, pages 1–71, 2006.

[156] Thang N. Dinh and My T. Thai. AI and Blockchain: A Disruptive Integration. *Computer*, 2018.

[157] Yueyue Dai, Du Xu, Sabita Maharjan, Zhuang Chen, Qian He, and Yan Zhang. Blockchain and Deep Reinforcement Learning Empowered Intelligent 5G beyond. *IEEE Network*, 2019.

[158] Khaled Salah, M. Habib Ur Rehman, Nishara Nizamuddin, and Ala Al-Fuqaha. Blockchain for AI: Review and open research challenges. *IEEE Access*, 2019.

[159] Charles Keating, Ralph Rogers, Resit Unal, David Dryer, Andres Sousa-Poza, Robert Safford, William Peterson, and Ghaith Rabadi. System of systems engineering. *EMJ - Engineering Management Journal*, 2003.

[160] Daniel A. DeLaurentis. Understanding transportation as system-of-systems design problem. In *43rd AIAA Aerospace Sciences Meeting and Exhibit - Meeting Papers*, 2005.

[161] Andrew Sage and Christopher Cuppan. On the Systems Engineering and Management of Systems of Systems and Federations of Systems. *Information Knowledge Systems Management*, 2001.

[162] Michael O. Ball, Chien-Yu Chen, Robert Hoffman, and Thomas Vossen. Collaborative Decision Making in Air Traffic Management: Current and Future Research Directions. 2001.

[163] Douglas Fearing and Cynthia Barnhart. Evaluating air traffic flow management in a collaborative decision-making environment. *Transportation Research Record*, 2011.

[164] Ahmad I. Jarrah, Jon Goodstein, and Ram Narasimhan. An Efficient Airline Re-Fleeting Model for the Incremental Modification of Planned Fleet Assignments. *Transportation Science*, 34(4):349–363, 2000.

[165] Hanif D. Sherali, Ebru K. Bish, and Xiaomei Zhu. Airline fleet assignment concepts, models, and algorithms. *European Journal of Operational Research*, 172(1):1–30, 2006.

[166] Oriol Lordan, Jose M. Sallan, Nuria Escorihuela, and David Gonzalez-Prieto. Robustness of airline route networks. *Physica A: Statistical Mechanics and its Applications*, 2016.

[167] Daniele Magazzeni, Peter Mcburney, and William Nash. Validation and verification of smart contracts: A research agenda. *Computer*, 2017.

[168] Federico Matteo Benčić and Ivana Podnar Žarko. Distributed Ledger Technology: Blockchain Compared to Directed Acyclic Graph. In *Proceedings - International Conference on Distributed Computing Systems*, 2018.

[169] Tyler J. Vanderweele and James M. Robins. Directed acyclic graphs, sufficient causes, and the properties of conditioning on a common effect. *American Journal of Epidemiology*, 2007.

[170] Brian Karrer and M. E.J. Newman. Random graph models for directed acyclic networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 2009.

[171] Serguei Popov. IOTA Whitepaper v1.4.3. *New Yorker*, 2018.

[172] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 1957.

[173] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. 2005.

[174] Stephane Bratu and Cynthia Barnhart. An Analysis of Passenger Delays Using Flight Operations and Passenger Booking Data. *Air Traffic Control Quartely*, 13(1):1–27, 2005.

[175] Stephen J. Maher. A novel passenger recovery approach for the integrated airline recovery problem. *Computers and Operations Research*, 57:123–137, 2015.

[176] Masahiro Murakawa, Toshio Adachi, Yoshihiro Niino, Yuji Kasai, Eiichi Takahashi, Kaoru Takasuka, and Tetsuya Higuchi. An AI-calibrated IF filter: A yield enhancement method with area and power dissipation reductions. *IEEE Journal of Solid-State Circuits*, 2003.

[177] Stephan Lewandowsky, Michael Mundy, and Gerard P.A. Tan. The dynamics of trust: Comparing humans to automation. *Journal of Experimental Psychology: Applied*, 2000.

[178] Kazuo Okamura and Seiji Yamada. Adaptive trust calibration for human-AI collaboration. *PLoS ONE*, 2020.

[179] Enrique Vidal, Franck Thollard, Colin de la Higuera, Francisco Casacuberta, and Rafael C. Carrasco. Probabilistic finite-state machines - Part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

[180] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence, 2015.

[181] Moray Allan and Christopher K. Christopher. Harmonising chorales by probabilistic inference. In *Advances in Neural Information Processing Systems*, 2005.

[182] Todd K. Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 1996.

[183] Chuong B. Do and Serafim Batzoglou. What is the expectation maximization algorithm? *Nature Biotechnology*, 2008.

[184] Paul Munro, Hannu Toivonen, Geoffrey I. Webb, Wray Buntine, Peter Orbanz, Yee Whye Teh, Pascal Poupart, Claude Sammut, Caude Sammut, Hendrik Blockeel, Dev Rajnarayan, David Wolpert, Wulfram Gerstner, C. David Page, Sriraam Natarajan, and Geoffrey Hinton. Baum-Welch Algorithm. In *Encyclopedia of Machine Learning*. 2011.

[185] David L. Poole and Alan K. Mackworth. *Artificial intelligence: Foundations of computational agents*. 2010.

[186] Michael Ball, Cynthia Barnhart, Martin Dresner, Mark Hansen, Kevin Neels, Amedeo Odoni, Everett Peterson, Lance Sherry, Antonio A Trani, and Bo Zou. Total delay impact study: a comprehensive assessment of the costs and impacts of flight delay in the United States. *The national center of excellence NEXTOR*, (December):2015, 2010.

[187] Felix Lau, Stuart H. Rubin, Michael H. Smith, and Ljiljana Trajković. Distributed denial of service attacks. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2000.

[188] Saman Taghavi Zargar, James Joshi, and David Tipper. A survey of defense mechanisms against distributed denial of service (DDOS) flooding attacks. *IEEE Communications Surveys and Tutorials*, 2013.

[189] Xu Hao, Long Yu, Liu Zhiqiang, Liu Zhen, and Gu Dawu. Dynamic practical byzantine fault tolerance. In *2018 IEEE Conference on Communications and Network Security, CNS 2018*, 2018.

[190] G. R. Blakley. Safeguarding cryptographic keys. In *1979 International Workshop on Managing Requirements Knowledge, MARK 1979*, 1979.

[191] Miguel Correia, Giuliana Santos Veronese, Nuno Ferreira Neves, and Paulo Verissimo. Byzantine consensus in asynchronous message-passing systems: A survey. *International Journal of Critical Computer-Based Systems*, 2011.

[192] H Ibrahim. Py-swirld: Swirlds Consensus Algorithm in Python, 2017.

[193] Elaine B Barker and Allen Roginsky. Recommendation for Cryptographic Key Generation. *NIST Special Publication 800-133*, 2012.

[194] Saurabh Sinha. On counting position weight matrix matches in a sequence, with application to discriminative motif finding. In *Bioinformatics*, 2006.

[195] Xuhua Xia. Position Weight Matrix, Gibbs Sampler, and the Associated Significance Tests in Motif Characterization and Prediction. *Scientifica*, 2012.

[196] Airlines for America. U.S. Passenger Carrier Delay Costs, 2019.

[197] R. Neal. Bayesian Learning for Neural Networks. *LECTURE NOTES IN STATISTICS -NEW YORK- SPRINGER VERLAG-*, 1996.

[198] David Gustafson. Schaum's Outline of Software Engineering. In *Theory and Problems of Software Engineering*, chapter 10, pages 145 – 156. McGraw-Hill, 1st edition, 2002.

[199] Ian Sommerville. Software Engineering: Principles and Practice. *Software Engineering Journal*, 1994.

[200] Nicholas R. Jennings. On agent-based software engineering. *Artificial Intelligence*, 2000.

APPENDICES

A. Nomenclature for Chapters 3, 4 & 5

A.1 Nomenclature for determinate aleatoric data features

Aleatoric Data Feature	**Description**	**Observation Input Category**
dow	Day of the week	FREQ
doy	Day of the year	FREQ
dest_x_dir	Destination airport location in spherical X coordinate	DEST
dest_y_dir	Destination airport location in spherical Y coordinate	DEST
dest_z_dir	Destination airport location in spherical Z coordinate	DEST
moy	Month of the year	FREQ
ONBD_CT	Total number of passengers onboard flight	PAX DMD
orig_x_dir	Origin airport location in spherical X coordinate	ORIG
orig_y_dir	Origin airport location in spherical Y coordinate	ORIG
orig_z_dir	Origin airport location in spherical Z coordinate	ORIG
route	Spherical distance between origin and destination airports	RTE

sched_route_originator_flag	Flag to indicate first flight of the day	ORIG
season	Season of the year	FREQ

A.2 Nomenclature for indeterminate aleatoric data features

Aleatoric Data Feature	**Description**	**Observation Input Category**	**Functional Role**
HD04	Holding aircraft for inbound connections	DISRP	Customer Hold
HD05	Holding aircraft to accommodate customers of delayed flight	DISRP	Customer Hold
HD10	Team accommodation	DISRP	Customer Hold
DP01	Flight dispatch activities	DISRP	Dispatch CSC
HD11	Holding flight to accommodate international schedule slot times	DISRP	Dispatch CSC
FO01	Pilot activities as they relate to normal aircraft readiness	DISRP	Flight Operations
FO02	Flight dispatch activities	DISRP	Flight Operations
FO03	Pilot scheduling activities	DISRP	Flight Operations

FO04	Pilot late for flight or missing items required for flight	DISRP	Flight Operations
EA11	Fueler late to aircraft	DISRP	Fuel Management
EA12	Fueling equipment failure	DISRP	Fuel Management
EA13	Inoperative gauges or dripstick	DISRP	Fuel Management
EA14	Any other energy administration delay	DISRP	Fuel Management
EA15	Heavy fuel load ($\geq$ 3000 gallons)	DISRP	Fuel Management
EA16	Uplift change	DISRP	Fuel Management
OP01	Customer boarding	DISRP	Ground Operations
OP02	Customer service	DISRP	Ground Operations
OP03	Ramp service	DISRP	Ground Operations
OP04	Ramp service - cargo	DISRP	Ground Operations
OP05	Provisioning - non provisioning station	DISRP	Ground Operations
OP06	Non routine	DISRP	Ground Operations
OP07	Southwest Airlines security	DISRP	Ground Operations
OP09	Ramp service delay - mail	DISRP	Ground Operations
OP11	Airstart - delay $> 5mins$	DISRP	Ground Operations
OP12	Towing aircraft	DISRP	Ground Operations
OP13	Operations	DISRP	Ground Operations

PV01	Provisioning - provisioning station activities	DISRP	Ground Operations
IF01	Flight attendant activities related to normal aircraft readiness	DISRP	Inflight
IF02	Flight attendant sick or injured online	DISRP	Inflight
IF03	Flight attendant scheduling activities	DISRP	Inflight
IF04	Flight attendant late for flight / missing items required for flight	DISRP	Inflight
MX01	Scheduled or non-scheduled inspection	DISRP	Maintenance
MX02	Mechanical problem	DISRP	Maintenance
MX03	Waiting on aircraft parts or mechanics	DISRP	Maintenance
MX09	Airstart - delay $< 5mins$	DISRP	Maintenance
HD01	ATC gate hold for congestion at departure station	DISRP	NAS

HD02	ATC gate hold for flow control enroute or at destination station	DISRP	NAS
MX06	Bird strike - maintenance inspection or aircraft swap	DISRP	NAS
SD01	Baggage screening by TSA at skycap or ticket counter	DISRP	Security
SD02	Excessive security checkpoint processing times	DISRP	Security
SD03	Selectee gate screening delay	DISRP	Security
SD04	Inoperative security screening equipment delay	DISRP	Security
SD05	Terminal or concourse security evacuation delay	DISRP	Security
SD06	Bomb threat	DISRP	Security
SD07	Weapons confiscation	DISRP	Security
SD08	Other TSA delay	DISRP	Security
SD09	Random TSA gate screening	DISRP	Security

SD10	Immigration and customs	DISRP	Security
SD11	Positive passenger bag match (PPBM)	DISRP	Security
OP10	Technology system outage	DISRP	Technology
HD03	Weather holding	DISRP	Weather
HD06	ATC gate hold for weather at departure station	DISRP	Weather
HD07	ATC gate hold for weather at enroute or at destination station	DISRP	Weather
HD08	Ice on wings / cold-soaked fuel	DISRP	Weather
HD09	Deicing at gate	DISRP	Weather
MX05	Inspection due to lightning strike	DISRP	Weather
MX07	Inspection due to turbulence	DISRP	Weather
MXO8	Hail ice, or snow damage	DISRP	Weather

A.3 Nomenclature for epistemic data features

Epistemic Data Feature	**Description**	**Activity Phase in UTFM**
ACTL_ACFT_TYPE	Actual aircraft type used	TAO

actl_block_mins	Actual blocktime period	TOO, EO, TIO
actl_enroute_mins	Actual flight period in the air	EO
ACTL_TURN_MINS	Actual turnaround period	TAO
ADJST_TURN_MINS	Adjusted turnaround period	TAD
DELY_MINS	Total delay period before actual pushback	TAD, TOD
DOT_DELAY_MINS	Total arrival delay	ED, TID
late_out_vs_sched_mins	Total departure delay	TOD
SCHED_ACFT_TYPE	Scheduled aircraft type used	TAS
sched_block_mins	Scheduled blocktime period	TOS, ES, TIS
SCHED_TURN_MINS	Scheduled turnaround period	TAS
shiftper_actl_GP	% work shift completed at actual gate parking time	TID
shiftper_actl_LD	% work shift completed at actual landing time	ED
shiftper_actl_PB	% work shift completed at actual pushback time	TOD
shiftper_actl_TO	% work shift completed at actual takeoff time	ED
shiftper_sched_GP	% work shift completed at scheduled gate parking time	TID
shiftper_sched_PB	% work shift completed at scheduled pushback time	TAD
SWAP_FLT_FLAG	Flight swap flag	TAS, TAD, TAO
taxi_in	Taxi-in period	TIS, TIO
taxi_out	Taxi-out period	TOS, TOO
tod_actl_GP	Actual aircraft gate parking time at destination	TIO

tod_actl_LD	Actual aircraft landing time at destination	EO
tod_actl_PB	Actual aircraft pushback time at origin	TAO
tod_actl_TO	Actual aircraft takeoff time at origin	TOO
tod_sched_GP	Scheduled aircraft gate parking time at destination	TIS
tod_sched_PB	Scheduled aircraft pushback time at origin	TAS

B. Algorithms for Chapters 4 & 5

B.1 Dynamic Programming Algorithm 1

Algorithm 1: Baum-Welch Algorithm

0: **procedure** BAUMWELCH(Y, X)

1: $A, B, \alpha, \beta \in Y$

2: **for** $t = 1 : N$ **do**

3: $\gamma(:, t) = \dfrac{\alpha(:, t) \odot \beta(:, t)}{\sum(\alpha(:, t) \odot \beta(:, t))}$

4: $\xi(:, :, t) = \dfrac{(\alpha(:, t) \odot A(t+1)) * (\beta(; , t+1) \odot B(X_{t+1}))^T}{\sum(\alpha(:, t) \odot \beta(:, t))}$

5: **end for** where $N = |X|$

6: $\hat{\pi} = \dfrac{\gamma(:, 1)}{\sum(\gamma(:, 1))}$

7: **for** $j = 1 : K$ **do**

8: $\hat{A}(j, :) = \dfrac{\sum(\xi(2 : N, j, :), 1)}{\sum(\sum(\xi(2 : N, j, :), 1), 2)}$

9: $\hat{B}(j, :) = \dfrac{X(:, j)^T \gamma}{\sum(\gamma, 1)}$

10: **end for** where K is number of states

11: **return** $\hat{\pi}, \hat{A}, \hat{B}$

11: **end procedure**=0

B.2 Dynamic Programming Algorithm 2

Algorithm 2: Viterbi Algorithm

```
0:  procedure VITERBI(Y, X)
1:  A, B, π ∈ Y
2:  Initialize:  δ_1 = π ∘ B_{X_1},  a_1 = 0
3:  for t = 2 : N do
4:      for j = 1 : K do
5:          [a_t(j), δ_t(j)] = max_i(log δ_{t-1}(:) + log A_{ij} + log B_{X_i}(j))
6:      end for   where K is number of states
7:  end for   where N = |X|
8:  Z*_N = arg max δ_N
9:  for t = N - 1 : 1 do
10:     Z*_t = a_{t+1} Z*_{t+1}
11: end for
12: return  Z*_{1:N}
12: end procedure=0
```

B.3 Dynamic Programming Algorithm 3

Algorithm 3: UTFM Learning Algorithm

0: **procedure** UTFMLearning(X, Y)

1: $X_S = \{s_1, ..., s_m\}$, $X_D = \{d_1, ..., d_m\}$, $X_O = \{o_1, ..., o_m\}$ Disrupted flight data

2: **for all** $j \in (1, 2, ..., m)$ **do**

3: $\mathcal{S}' \leftarrow S_j$, $\mathcal{D}' \leftarrow D_j$, $\mathcal{O}' \leftarrow O_j$,
$A' \leftarrow \alpha_{ij} : S_i \rightarrow S_j$, $B' \leftarrow \beta_{ij} : D_i \rightarrow D_j$, $\Gamma' \leftarrow \gamma_{ij} : O_i \rightarrow O_j$
$K' \leftarrow \kappa_j : S_j \rightarrow D_j$, $\Lambda' \leftarrow \lambda_j : D_j \rightarrow O_j$ for $i = j - 1$ and $i > 0$

4: $\mathcal{M}' \leftarrow \{\mathcal{S}', \mathcal{D}', \mathcal{O}', A', B', \Gamma', K', \Lambda'\}$ Initialize Optimal HMM sets for UTFM

5: **while** $|X_S|, |X_D|, |X_O| > m$ **or** $\neg\mathcal{M}'$ **do**

6: $Y_S = \{y_{s1}, ..., y_{sl}\}$, $Y_D = \{y_{d1}, ..., y_{dl}\}$,
$Y_O = \{y_{o1}, ..., y_{ol}\}$ Training (flight schedule) data for $l \geq m$

7: $S'_j \leftarrow$ BaumWelch(S_j, Y_S), $D'_j \leftarrow$ BaumWelch(D_j, Y_D),
$O'_j \leftarrow$ BaumWelch(O_j, Y_O),
$\alpha'_{ij} \leftarrow$ BaumWelch(α_{ij}, Y_S), $\beta'_{ij} \leftarrow$ BaumWelch(β_{ij}, Y_D),
$\gamma'_{ij} \leftarrow$ BaumWelch(γ_{ij}, Y_O),
$\kappa'_j \leftarrow$ BaumWelch(κ_j, Y_D), $\lambda'_j \leftarrow$ BaumWelch(λ_j, Y_O)

8: $\mathcal{S}' \leftarrow S'_j$, $\mathcal{D}' \leftarrow D'_j$, $\mathcal{O}' \leftarrow O'_j$, $A' \leftarrow \alpha'_{ij}$, $B' \leftarrow \beta'_{ij}$, $\Gamma' \leftarrow \gamma'_{ij}$,
$K' \leftarrow \kappa'_j$, $\Lambda' \leftarrow \lambda'_j$ Update Optimal HMM sets for UTFM

9: **end while**

10: **end for**

11: $\mathcal{N}_{\prime} \leftarrow \{\mathcal{S}', \mathcal{D}', \mathcal{O}'\}$, $\mathcal{N}_{\rangle\rightarrow|} \leftarrow \{A', B', \Gamma'\}$, $\mathcal{N}_{\rangle\rightarrow\rangle} \leftarrow \{K', \Lambda'\}$

12: $\mathcal{N}_{\rightarrow} \leftarrow \{\mathcal{N}_{\rangle\rightarrow\rangle}, \mathcal{N}_{\rangle\rightarrow|}\}$

13: $\mathcal{K} \leftarrow (\mathcal{N}_{\prime}, \mathcal{N}_{\rightarrow})$ Optimal (RDBN) Data Architecture for UTFM

13: **end procedure**=0

B.4 Dynamic Programming Algorithm 4

Algorithm 4: UTFM Decoding Algorithm

Require: $\mathcal{K}$ Optimal UTFM Architecture

0: **procedure** UTFMDECODING(X)

1: $P(s) \leftarrow \textsc{Viterbi}(\mathcal{S}', X_S)$, $P(d) \leftarrow \textsc{Viterbi}(\mathcal{D}', X_D)$,
$P(o) \leftarrow \textsc{Viterbi}(\mathcal{O}', X_O)$,
$P(\alpha) \leftarrow \textsc{Viterbi}(A', X_S)$, $P(\beta) \leftarrow \textsc{Viterbi}(B', X_D)$,
$P(\gamma) \leftarrow \textsc{Viterbi}(\Gamma', X_O)$,
$P(\kappa) \leftarrow \textsc{Viterbi}(K', X_D)$, $P(\lambda) \leftarrow \textsc{Viterbi}(\Lambda', X_O)$ Unroll $\mathcal{K}$ with disrupted flight information

2: **for all** $j \in (1, 2, ..., m)$ **do**

3: $\phi_j \leftarrow P(s_j) + P(\alpha_{ij}) + P(\kappa_j)$, $\psi_j \leftarrow P(d_j) + P(\beta_{ij}) + P(\lambda_j)$,
$\rho_j \leftarrow P(o_j) + P(\gamma_{ij})$
for $i = j - 1$ and $i > 0$

4: $a \leftarrow \frac{P(s_j)}{\phi_j}$, $b \leftarrow \frac{P(\alpha_{ij})}{\phi_j}$, $c \leftarrow \frac{P(\kappa_j)}{\phi_j}$, $p \leftarrow \frac{P(d_j)}{\psi_j}$, $q \leftarrow \frac{P(\beta_{ij})}{\psi_j}$, $r \leftarrow \frac{P(\lambda_j)}{\psi_j}$
$u \leftarrow \frac{P(o_j)}{\rho_j}$, $v \leftarrow \frac{P(\gamma_{ij})}{\rho_j}$ State probabilities (stochastic matrix) for UTFM

5: **end for**

6: $N_0 \leftarrow \{a, p, u\}$, $N_{i \to j} \leftarrow \{b, q, v\}$, $N_{i \to i} \leftarrow \{c, r\}$

7: $N_{\to} \leftarrow \{N_{i \to i}, N_{i \to j}\}$

8: $K \leftarrow (N_0, N_{\to})$ UTFM for disrupted flight

9: **return** K

9: **end procedure**=0

C. Single-Layer Neural Network as Gaussian Process

Let the ith component of a single-layer neural network output z_i^1 be computed as:

$$z_i^1 = b_i^1 + \sum_{j=1}^{N_1} W_{ij}^1 x_j^1(x), \quad x_j^1(x) = f(b_j^0 + \sum_{k=1}^{d_{in}} W_{jk}^0 x_k) \tag{C.1}$$

where x represents the input, W and b represent the weight and bias parameters at appropriate layers respectively, and N_1 represents the width of the single hidden layer. Since the weight and bias parameters are set to be independent and identically distributed (i.e. i.i.d.), thus applying the Central Limit Theorem (CLT) ensures that z_i^1 will be Gaussian distributed if and only if $N_1 \to \infty$. Without loss of specificity, any finite set of outputs $\{z_i^1(x^{\beta=1}), ..., z_i^1(x^{\beta=k})\}$ that follows the multidimensional CLT will have a joint multivariate Gaussian distribution, which defines a Gaussian process. As such, $z_i^1 \sim \mathcal{GP}(\mu^1, K^1)$, where μ^1 and K^1 represent the mean and covariance parameters of the Gaussian process, respectively, and are independent of i. Thus, for $\mu^1(x) = \mathbb{E}[z_i^1(x)] = 0$ and,

$$K^1(x, x') \simeq \mathbb{E}[z_i^1(x) z_i^1(x')] = \sigma_b^2 + \sigma_w^2 \mathbb{E}[x_i^1(x) x_i^1(x')] \simeq \sigma_b^2 + \sigma_w^2 Q(x, x') \tag{C.2}$$

where $Q(x, x')$ is retrieved by summing the distribution of W^0, b^0 along infinitely many discrete intervals. For $i \neq j$, any two outputs, z_i^1, z_j^1, of the single-layer network are guaranteed to be joint Gaussian with zero covariance. Refer to [126, 197] for more information and proofs for relationships between Gaussian processes and neural networks.

D. Testing Methods for Algorithms and Processes

The following framework and approaches for software testing and verification are adapted from Schaum's Outline of Software Engineering [198]. As such, the deployment of the algorithms and processes (i.e. software) that define the *i-MAS* framework can be separated into two categories of software testing, shown in Fig. D.1, namely: static testing and dynamic testing.

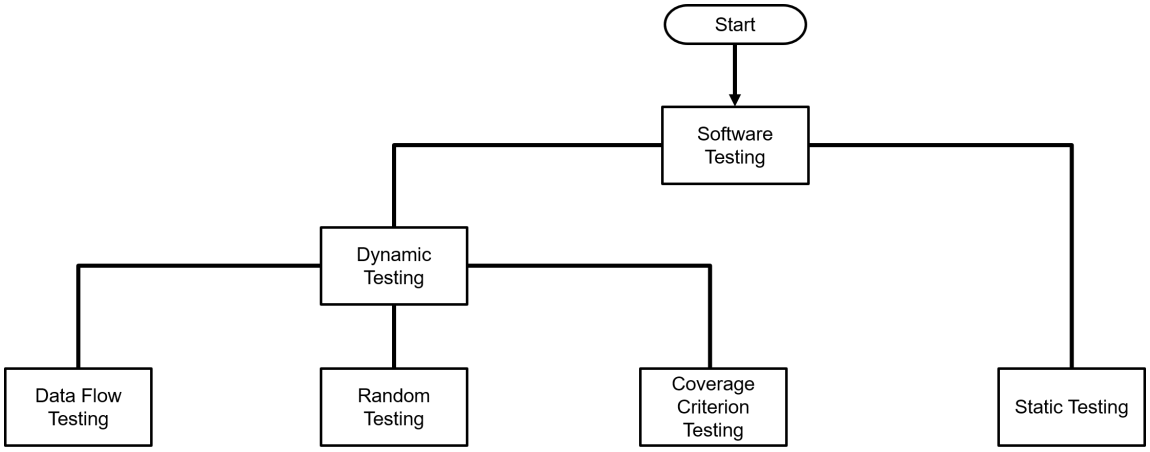

Figure D.1. Software testing routine for *i-MAS* algorithms and processes

D.1 Static Testing

Static testing verifies that basic mathematical functions and necessary packages, for the programming languages in which the *i-MAS* architecture is written, are valid and produce the appropriate outcomes. As such, we assume that the developers of many fundamental packages for Python, which represents the programming language used for the development of the *i-MAS* platform discussed in this work, conducted extensive static testing before their packages were deployed to the corresponding

Python libraries. To that effect, we performed minimal static testing for the development of the *i-MAS* architecture and focus the scope of our software testing on dynamic testing.

D.2 Dynamic Testing

Ideally, dynamic testing of the algorithms and processes for the *i-MAS* architecture represents the execution of every possible test scenario for the decentralized AI platform with actual test data. However, it is practically infeasible to perform exhaustive testing for the deployment of the *i-MAS* architecture because there are too many possible test cases due to the platform's complexity and infinitely long testing duration [198]. Thus, the main objective of dynamic testing is to execute a very minute percentage of possible test cases, by adequately discerning what test cases to use and how many of these cases are necessary before deployment [199, 200]. Human specialists across multiple domains in the AOCC can provide a plethora of input criteria for the *i-MAS* platform without any definitive ascertainment that their inputs would result in the perfect recovery plan. As such, the decentralized AI platform (i.e. *i-MAS*) is fundamentally agnostic. To this effect, we employ a combination of several forms of dynamic testing methods, shown in Fig. D.1 and discussed in subsequent sections, to ensure that major components of the *i-MAS* architecture can be properly tested and deployed in reasonable time.

D.2.1 Data Flow Testing

Data flow testing represents methods for ensuring the movement of input data from a human specialist through the *i-MAS* platform. The definition of data indicates the instance when a specific value, either from a human specialist or historical record, is assigned to a particular data feature. As such, a path from the definition of a data feature from historical record to a use of that data feature by a human specialist, which does not involve another definition of the data feature from historical record,

represents a *definition free path*. There are two separate methods for the use of any defined data feature through a *definition-free path* in the *i-MAS* platform, namely: *computation use* and *predicate use*.

- ***Computation-use*** for *i-MAS* represents data flow test situations where a defined data feature appears on the right-hand side of an assignment statement. To this effect, we apply a criteria that requires a *definition-free* path from every possible definition of data features to a *computation-use*. As such, all inference and decoding methods for different components of the UTFM and PTFM are subjected to *computation-use* for data flow testing, by ensuring that appropriate mathematical and statistical formulations are satisfied. For instance, a characteristic state transition matrix for the UTFM, retrieved by unrolling an optimal set of HMMs with refined data features from a human specialist via the Viterbi algorithm, must satisfy the properties of a stochastic matrix. As such, the values of columns and/or rows in the stochastic matrix from a UTFM decoding, for any instantiation of test data, are always guaranteed to sum to one, as shown in Fig. 4.10 and Fig. 4.11.
- ***Predicate-use*** represents data flow test situations where a defined data feature from a human specialist is used as the condition of a decision statement by the *i-MAS* framework, such that a *predicate-use* can be imposed on both branches of the decision statement. Thus, we apply a *predicate-use* criteria, which requires a *definition-free-path* from every possible definition to a *predicate-use*, for all learning and aggregating methods for different components of the UTFM and PTFM. For example, prior to instantiating the UTFM and PTFM in the *i-MAS* platform, the cardinality of all necessary data features for each component of the UTFM and PTFM are checked to ensure that the most optimal versions of the UTFM and PTFM are invoked. The cardinality represents the number of significant data features appropriate for defining the behavior of UTFM and PTFM components, which describe multiple separate phases of airline disruption management.

D.2.2 Random Testing

Random testing represents a method where random data samples are selected to facilitate statistical inference and verify the efficacy of statistical assumptions for different components of the *i-MAS* platform. As such, all data features for exercising different components of the UTFM and PTFM are initialized by applying pseudorandom number generators, which automatically and randomly select different instances of training data and test data for developing and validating the respective models. For instance, to develop the PTFM, we employ 70% of the total available data to train its component ANN models and the remaining 30% of the data to test the statistical validity of ANN models, while applying a random seed of 42 to initialize pseudorandom selection of training and test data.

D.2.3 Coverage Criterion Testing

Coverage criterion testing represents a testing method for creating rules on how to choose tests and when to stop testing for systems that define the *i-MAS* platform before deployment. To this effect, we tested the PTFM, which represents the brain of the *i-MAS* platform, by ensuring that the ANN models for separate parts of the PTFM are trained to meet specific criteria acceptable for airline disruption management. For instance, we employed the root mean square error (RMSE), which defines the standard deviation of unexplained variance, to adjudicate the effectiveness of generic ANN models for the PTFM by ensuring that the RMSE for turnaround duration or block time duration was less than equal to $14mins$. The threshold value of $14mins$ was selected based upon current industry standards for tardiness in flight schedule arrival, such that any flight that arrives within $14mins$ of the original scheduled time during disruption management is considered to have arrived on time. As such, by appropriating a test data set representative of 30% of the total available data through regression analysis, we ensured that all ANN models for estimating turnaround duration and block time duration achieved a RMSE value that is less

than $14mins$, as shown in Table 4.5. For the UTFM, we applied a cross-validation of the total sum of log-likelihoods of separately partitioned data samples to define the test coverage criteria, while learning the optimal parameters for a generic HMM representing a UTFM state.

VITA

VITA

See next pages for curriculum vitae.

Relevant Industry Experience

Southwest Airlines Dallas, TX

Flight Operations Engineering Intern Summer 2017, Summer 2013, Fall 2012

- Designed new and revamped existing One Engine Inoperative (OEI) departure flight procedures to achieve the highest plausible Maximum Takeoff Weight (MTOW) from major and minor Southwest Airlines stations. Procedures validated in hydraulic-enabled Boeing 737 simulators and published in Jeppesen 10-7A charts
- Developed an application using FORTRAN to manage Boeing Airplane Performance Monitoring (APM) data for Boeing 737-300,500,700,800. Data used for real-time fuel bias analysis of Southwest Airlines Boeing 737 aircraft fleet
- Assisted with implementation of the Electronic Flight Bag (EFB) program at Southwest Airlines
- Evaluated the impact of existing and proposed obstacles to support management decision in airport negotiations and obstacle mitigation
- Collaborated with operations engineers on the audit and maintenance of the aircraft onboard performance system (OPS) to support current Southwest Airlines operations, while verifying, validating, and integrating the OPS successor

Southwest Airlines Dallas, TX

Powerplant Operations Engineering Intern Summer 2016

- Proactively reviewed daily oil consumption rates of all jet engines in the Southwest Airlines fleet to ensure compliance with guidelines provided by the engine manufacturer
- Issued appropriate work instructions to aircraft maintenance technicians for oil consumption related repairs on affected engines
- Investigated the correlation between different fuel nozzle types in the engine fleet and exhaust gas temperature (EGT) indication incidents recorded in GE customer notification reports (CNR)

Publications

[1] S. Jain, K. E. Ogunsina, H. Chao, W. A. Crossley, and D. A. DeLaurentis, "Predicting Routes for, Number of Operations of, and Fleet-level Impacts of Future Commercial Supersonic Aircraft on Routes Touching the United States", 2020.

[2] K. E. Ogunsina, M. Papamichalis, I. Bilionis, and D. A. DeLaurentis, "Hidden Markov Models for Pattern Learning and Recognition in a Data-Driven Model for Airline Disruption Management", 2019.

[3] **K. Ogunsina**, H. Chao, N. Kolencherry, S. Jain, K. Moolchandani, W. Crossley, and D. DeLaurentis, "Fleet-level Environmental Assessments for Feasibility of Aviation Emission Reduction Goals", in *CESUN 2018 International Engineering Systems Symposium*, 2019.

[4] **K. Ogunsina**, N. Davendralingram, I. Bilionis, and D. Delaurentis, "Dimensionality reduction in a data-driven model for airline disruption management", in *AIAA Scitech 2019 Forum*, 2019, ISBN: 9781624105784.

[5] **K. Ogunsina**, H. Chao, N. Kolencherry, K. Moolchandani, W. A. Crossley, and D. A. DeLaurentis, "A model of aircraft retirement and acquisition decisions based on net present value calculations", in *17th AIAA Aviation Technology, Integration, and Operations Conference, 2017*, 2017, ISBN: 9781624105081.

[6] J. Behmer, **K. Ogunsina**, P. Shah, and S. Srinivasan, "An agent-based modeling approach to creating more resilient littoral combat architectures", in *IEEE Aerospace Conference Proceedings*, 2016, ISBN: 9781467376761.

Academic Work Experience

- **Graduate Research Assistant** at Purdue University — January 2015 - December 2020
  FAA ASCENT Project 010, FAA NEXTOR II
- **Graduate Teaching Assistant** at Purdue University — January 2015 - May 2017
  Control Systems Laboratory, Introduction to Aerospace Design, Aeromechanics

Skills

- **Software:** MATLAB, Simulink, FLOPS, Python, FORTRAN 90/95, CATIA v5, NASTRAN, Digital Datcom, XML, VLAERO+, Surfaces
- **Training:** Lean Methodology, 6-Sigma, Safety Management System (SMS)

Languages

- **Languages:** English (fluent), Yoruba (fluent)

Scholarships and Awards

- Outstanding Performance Award, Southwest Airlines — December 2012
- Outstanding Performance Award, Southwest Airlines — August 2013
- Undergraduate Outstanding Student of the Year, ERAU — April 2014
- Outstanding Performance Award, Southwest Airlines — August 2017
- Bilsland Dissertation Fellowship, Purdue University — August 2019 - August 2020