An Analysis of the Effects of Net-Centric Operations Using Multi-Agent Adaptive Behavior

Calderon-Meza, Guillermo

http://hdl.handle.net/1920/6362

a service of Mason Publishing
AN ANALYSIS OF THE EFFECTS OF NET-CENTRIC OPERATIONS USING MULTI-AGENT ADAPTIVE BEHAVIOR

by

Guillermo Calderón-Meza
A Dissertation
Submitted to the Graduate Faculty of George Mason University in Partial Fulfillment of The Requirements for the Degree of Doctor of Philosophy Information Technology

Committee:

Dr. Lance Sherry, Dissertation Director
Dr. Robert Axtell, Committee Member
Dr. Kenneth De Jong, Committee Member
Dr. Thomas Speller Jr., Committee Member
Dr. Daniel Menascé, Senior Associate Dean
Dr. Lloyd J. Griffiths, Dean, Volgenau School of Engineering

Date: 4/27/11

Spring Semester 2011
George Mason University
Fairfax, VA
An Analysis of the Effects of Net-Centric Operations Using Multi-Agent Adaptive Behavior

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Guillermo Calderón-Meza
Master of Science in Electronic Systems
University of Bolton, UK / Universität Paderborn, Germany, 2000
Bachelor of Science in Electronics
Instituto Tecnológico de Costa Rica, Costa Rica, 1994

Director: Dr. Lance Sherry, Professor
Volgenau School of Engineering

Spring Semester 2011
George Mason University
Fairfax, VA
Dedication

I dedicate this dissertation to God who made everything possible, and to Maricel, Alejandro, Andrea, and Ariana who sacrificed with me all these years.
Dr. Sherry served as director of this dissertation. He was a true mentor and friend, without which this dissertation could not have been completed. His insistence on conducting research on the adaptive nature of the National Airspace System has proved to be visionary and has moved the field to new plateau. His knowledge of the air transportation domain and system engineering was invaluable.

Dr. De Jong provided insights into the scientific method and problem formulation that was instrumental in shaping this research. His knowledge on evolutionary computing and the interpretation of agent-based simulation results was critical.

Dr. Axtell, an internationally renowned expert on agent-based modeling, provided philosophical and technical guidance on the design, implementation, and testing of agent-based models. He challenged me with some of the toughest questions. I am grateful for his assistance.

Dr. Speller provided an outsiders viewpoint on this research. His deep knowledge of System Engineering and the scientific method were instrumental in shaping the research and generating clear, understandable descriptions of all aspects of the research.

In addition, there were several others who contributed to the research: Natalia Alexandrov, Banavar Sridhar, Kapil Sheth, María Consiglio, Brian Baxley, Kurt Neitzke, and Shon Grabbe (NASA), Joe Post, Kimberly Noonan, and Tony Diana (FAA), George Hunter and Huina Gao (Sensis), Jason Burke, Dr. Terry Thompson, Bob Hoffman (Metron Aviation), Fred Wieland (Intelligent Automation Inc.), Dr. George Donohue, Dr. John Shortle, Dr. Rajesh Ganesan, and Dr. Paulo Costa (George Mason University), John Ferguson, Keith Sullivan, Maciej Latek, Maricel Medina-Mora (George Mason University).

I want to thank the Ministerio de Ciencia y Tecnología (Minister of Science and Technology) of Costa Rica for providing financial support.

Finally, I thank Gary and Marilyn Rossen for their invaluable help and support.

This research was funded by The National Aeronautics and Space Administration (NASA) NRA NNA07CN32A, and internal research funding from the Center for Air Transportation Research (CATSR) housed at George Mason University (GMU).
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xi</td>
</tr>
<tr>
<td>List of Abbreviations</td>
<td>xv</td>
</tr>
<tr>
<td>Abstract</td>
<td>xviii</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 The National Airspace System</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1 Structure of the NAS</td>
<td>2</td>
</tr>
<tr>
<td>1.1.2 NAS Performance</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Next Generation Air Transportation System, NextGen</td>
<td>4</td>
</tr>
<tr>
<td>1.3 NAS-Wide Simulators and Modernization Benefits Analysis</td>
<td>6</td>
</tr>
<tr>
<td>1.4 Problem Statement</td>
<td>7</td>
</tr>
<tr>
<td>1.5 Objective</td>
<td>7</td>
</tr>
<tr>
<td>1.6 Research Approach</td>
<td>8</td>
</tr>
<tr>
<td>1.7 Summary of Results</td>
<td>9</td>
</tr>
<tr>
<td>1.8 Contributions</td>
<td>12</td>
</tr>
<tr>
<td>1.9 Benefits</td>
<td>12</td>
</tr>
<tr>
<td>2 Background and Literature Review</td>
<td>13</td>
</tr>
<tr>
<td>2.1 The National Airspace System</td>
<td>13</td>
</tr>
<tr>
<td>2.1.1 Navigation Aids and Airways</td>
<td>14</td>
</tr>
<tr>
<td>2.1.2 Airspace and Flight Rules</td>
<td>15</td>
</tr>
<tr>
<td>2.2 Next Generation Air Transportation System</td>
<td>16</td>
</tr>
<tr>
<td>2.3 Net-Centric Operations</td>
<td>18</td>
</tr>
<tr>
<td>2.3.1 Principles of NCO</td>
<td>20</td>
</tr>
<tr>
<td>2.3.2 Benefits from NCO to the NAS</td>
<td>22</td>
</tr>
<tr>
<td>2.3.3 Unintended Consequences of NCO in the NAS</td>
<td>24</td>
</tr>
<tr>
<td>2.3.4 Information Sharing and Adaptation in the NAS and NextGen</td>
<td>24</td>
</tr>
<tr>
<td>2.4 NAS-Wide Simulators</td>
<td>28</td>
</tr>
<tr>
<td>2.5 Cost and Benefit Evaluation of NAS Improvements</td>
<td>31</td>
</tr>
</tbody>
</table>
2.6 Theoretical Foundations ............................................. 33
   2.6.1 Agent-Based Models and Simulations .......................... 34
   2.6.2 Game Theory ................................................... 37
   2.6.3 Reinforcement Learning ....................................... 38
   2.6.4 Comparison of Performance Vectors ......................... 42
3 Methodology ................................................................. 45
   3.1 NAS-Wide Simulations with Adaptive Airline Decision-Making ........ 45
      3.1.1 A Particular Case: Pre-Departure Flightplan Route Selection ... 46
   3.2 Software Perspective .............................................. 48
      3.2.1 Static View of the Simulator ............................... 49
      3.2.2 Dynamic View of the Simulator ............................ 57
   3.3 The Database ....................................................... 59
   3.4 Algorithms for Main Application Component ..................... 68
      3.4.1 The Flight Specifications Input ............................ 68
      3.4.2 The Configuration File ..................................... 73
      3.4.3 Selection of Flightplan Routes ............................ 73
      3.4.4 Adaptation of Airline Decision-Making .................... 88
   3.5 Design of Experiments ........................................... 92
3.6 Analysis of Results .................................................. 97
4 Evaluation of the Impact of SWIM on the Performance of the NAS in the Presence of Adaptive Airline Flightplan Route Selection .............. 101
   4.1 Stability of the Learning Process ............................... 101
   4.2 Normality of the Data Samples ................................. 104
   4.3 Analyzing the Effects of Availability to Global Information .......... 106
   4.4 Analyzing the Effects of Latency with Global Information Available .... 118
   4.5 Analyzing the Effects of Latency with Local Information Available .... 128
   4.6 Analyzing the Effects of Inaccuracies in Global Information .......... 138
   4.7 Analyzing the Effects of Inaccuracies in Local Information .......... 146
5 Conclusions ................................................................. 157
   5.1 Summary of Results .............................................. 157
      5.1.1 Steady-State ................................................. 157
      5.1.2 The Exploration Period ..................................... 158
      5.1.3 Effect of the Independent Variables on System Performance ........ 158
      5.1.4 Effect of the Independent Variables on the Variance of the System Performance ............................................. 160
5.2 Discussion of Analysis ............................................. 160
   5.2.1 The Exploration Period ...................................... 161
   5.2.2 Effects of Global Information ............................... 163
   5.2.3 Effects of Latency in the Information ...................... 163
   5.2.4 Effects of Inaccuracies in the Information ................. 165
   5.2.5 Extensions to the Analysis ................................. 166
   5.2.6 Summary of contributions ................................. 169

5.3 Limitations and Future Work ................................. 169
   5.3.1 Performance Metrics ....................................... 170
   5.3.2 Improvements in the Computation of Rewards .............. 170
   5.3.3 Improvements to the Input File ............................ 171
   5.3.4 Compression of the Q-Matrix ............................... 171
   5.3.5 Stochasticity in NAS-Wide Simulation ..................... 171
   5.3.6 Alternate Routes .......................................... 172
   5.3.7 Economical Significance ................................. 173
   5.3.8 Collaborative Trajectory Options Program ................. 173
   5.3.9 Simulation Execution Time ................................ 173
   5.3.10 Validation of the Approach and the Results .............. 174

A Configuration Files and Scripts for Execution and Analysis .... 175
   A.1 SQL Scripts .................................................... 175
   A.2 Configuration File ............................................. 180
   A.3 Execution of a Simulation .................................... 182
   A.4 GNUPlot Scripts .............................................. 183
   A.5 R Scripts ...................................................... 192
      A.5.1 Script to Test for Normality of the Datasets ............ 192
      A.5.2 Script to Test for Equality of the Variances .......... 193
      A.5.3 Script to Test the Hypotheses for the Difference of Means 194
      A.5.4 Script to Test the Hypotheses for the Ratio of Variances 196

B Output Files of the Experiments ............................... 197
   B.1 The Flight Records File ...................................... 197
   B.2 The Sector Utilization File .................................. 198
   B.3 The Airport Arrivals File .................................. 199

C Java Source Code .............................................. 200
   C.1 FACETExperiment Class ...................................... 200
   C.2 FACETAgent Class ............................................ 206
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.3 Airline Classes</td>
<td>207</td>
</tr>
<tr>
<td>C.4 Flight Class</td>
<td>210</td>
</tr>
<tr>
<td>Bibliography</td>
<td>213</td>
</tr>
</tbody>
</table>
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>NAS-wide simulation tools, decision-making and adaptation functionality</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>Description of the airspace classes in the NAS</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>NextGen goals and objectives as described in the Concept of Operations for the Next Generation Air Transportation System v2.0</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>NAS-wide simulation tools, decision-making and adaptation functionality</td>
<td>29</td>
</tr>
<tr>
<td>2.4</td>
<td>Techniques to compare multiobjective performance metrics</td>
<td>43</td>
</tr>
<tr>
<td>3.1</td>
<td>Lines of code (LOC) for all the classes in the Main Application component</td>
<td>58</td>
</tr>
<tr>
<td>3.2</td>
<td>Data dictionary of the database</td>
<td>63</td>
</tr>
<tr>
<td>3.3</td>
<td>Description of the keys in the configuration file</td>
<td>75</td>
</tr>
<tr>
<td>3.4</td>
<td>Performance metrics used in the simulations</td>
<td>88</td>
</tr>
<tr>
<td>3.5</td>
<td>Design of experiments</td>
<td>95</td>
</tr>
<tr>
<td>3.6</td>
<td>VFR Airport Arrival Rates (AAR) used in the simulations for the OEP-35 airports</td>
<td>96</td>
</tr>
<tr>
<td>4.1</td>
<td>Number of simulated days needed to meet the stability criteria and data sample sizes</td>
<td>104</td>
</tr>
<tr>
<td>4.2</td>
<td>Results of the normality tests for the system metrics of all experiments using the Shapiro-Wilk’s test</td>
<td>105</td>
</tr>
<tr>
<td>4.3</td>
<td>Parameters for the F-test to determine equality of sample variances</td>
<td>115</td>
</tr>
<tr>
<td>4.4</td>
<td>Tests for equality of variances of the system-wide performance metrics when AOCs have global information vs. local information</td>
<td>115</td>
</tr>
<tr>
<td>4.5</td>
<td>Parameters of the t-tests to determine differences between the mean values</td>
<td>117</td>
</tr>
<tr>
<td>4.6</td>
<td>Hypothesis testing for the means of the system-wide metrics when AOCs have global information vs. local information</td>
<td>117</td>
</tr>
<tr>
<td>4.7</td>
<td>Hypothesis testing for the variances of the system-wide performance metrics when AOCs have global information vs. local information</td>
<td>118</td>
</tr>
<tr>
<td>4.8</td>
<td>Tests for equality of variances of the system-wide performance metrics when AOCs have delayed information vs. real-time information</td>
<td>127</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.9</td>
<td>Hypothesis testing for the means of the system-wide metrics when AOCs have global delayed information vs. global real-time information</td>
<td>127</td>
</tr>
<tr>
<td>4.10</td>
<td>Hypothesis testing for the variance of the system-wide performance metrics when AOCs have global delayed information vs. global real-time information</td>
<td>128</td>
</tr>
<tr>
<td>4.11</td>
<td>Tests for equality of variances of the system-wide performance metrics when AOCs have local delayed information vs. local real-time information</td>
<td>137</td>
</tr>
<tr>
<td>4.12</td>
<td>Hypothesis testing for the means of the system-wide metrics when AOCs have local delayed information vs. local real-time information</td>
<td>137</td>
</tr>
<tr>
<td>4.13</td>
<td>Hypothesis testing for the variance of the system-wide performance metrics when AOCs have local delayed information vs. local real-time information</td>
<td>138</td>
</tr>
<tr>
<td>4.14</td>
<td>Tests for equality of variances of the system-wide performance metrics when AOCs have global inaccurate information vs. global accurate information</td>
<td>147</td>
</tr>
<tr>
<td>4.15</td>
<td>Hypothesis testing for the means of the system-wide metrics when AOCs have global inaccurate information vs. global accurate information</td>
<td>147</td>
</tr>
<tr>
<td>4.16</td>
<td>Hypothesis testing for the variance of the system-wide performance metrics when AOCs have global inaccurate information vs. global accurate information</td>
<td>147</td>
</tr>
<tr>
<td>4.17</td>
<td>Tests for equality of variances of the system-wide performance metrics when AOCs have local inaccurate information vs. local accurate information</td>
<td>156</td>
</tr>
<tr>
<td>4.18</td>
<td>Hypothesis testing for the means of the system-wide metrics when AOCs have local inaccurate information vs. local accurate information</td>
<td>156</td>
</tr>
<tr>
<td>4.19</td>
<td>Hypothesis testing for the variance of the system-wide performance metrics when AOCs have local inaccurate information vs. local accurate information</td>
<td>156</td>
</tr>
<tr>
<td>5.1</td>
<td>Number of simulated days needed by the experiments to reach the steady-state</td>
<td>157</td>
</tr>
<tr>
<td>5.2</td>
<td>Reductions in the system-wide performance metrics between the pure random flightplan route selection and the start of the steady-state</td>
<td>159</td>
</tr>
<tr>
<td>5.3</td>
<td>Effect of the independent variable on the mean of the system-wide performance metrics</td>
<td>160</td>
</tr>
<tr>
<td>5.4</td>
<td>Effect of the independent variables on the variance of the system-wide performance metrics</td>
<td>161</td>
</tr>
<tr>
<td>5.5</td>
<td>Effect of the independent variables on the variance of the system-wide performance metrics</td>
<td>164</td>
</tr>
<tr>
<td>5.6</td>
<td>Dominant routes for each time of the day when AOCs have global, real-time, and accurate information</td>
<td>168</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Examples of actual flight plans and a direct route from LAX to JFK</td>
<td>3</td>
</tr>
<tr>
<td>2.1</td>
<td>Examples of actual flight plans and a direct route from LAX to JFK</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>Activity diagram of a CTOP program</td>
<td>27</td>
</tr>
<tr>
<td>3.1</td>
<td>Concept of NAS-wide simulation with adaptive stakeholder behavior</td>
<td>46</td>
</tr>
<tr>
<td>3.2</td>
<td>Examples of actual flight plans and a direct route from LAX to JFK</td>
<td>47</td>
</tr>
<tr>
<td>3.3</td>
<td>Distribution of flightplan route alternatives in the repository</td>
<td>47</td>
</tr>
<tr>
<td>3.4</td>
<td>UML component diagram of the simulator</td>
<td>48</td>
</tr>
<tr>
<td>3.5</td>
<td>Packages in the Main Application component</td>
<td>50</td>
</tr>
<tr>
<td>3.6</td>
<td>UML class diagram of the facet.experiments package</td>
<td>52</td>
</tr>
<tr>
<td>3.7</td>
<td>UML class diagram for the facet.experiments.parsers package</td>
<td>53</td>
</tr>
<tr>
<td>3.8</td>
<td>UML class diagram for the utils package</td>
<td>54</td>
</tr>
<tr>
<td>3.9</td>
<td>UML class diagram for the utils.databases package</td>
<td>55</td>
</tr>
<tr>
<td>3.10</td>
<td>UML class diagram for the utils.distributions package</td>
<td>56</td>
</tr>
<tr>
<td>3.11</td>
<td>UML class diagram for the utils.multiobjective package</td>
<td>56</td>
</tr>
<tr>
<td>3.12</td>
<td>UML sequence diagram for one execution of the simulator</td>
<td>60</td>
</tr>
<tr>
<td>3.13</td>
<td>UML state diagram for the Flight agent</td>
<td>61</td>
</tr>
<tr>
<td>3.14</td>
<td>Design of the database for the simulator</td>
<td>62</td>
</tr>
<tr>
<td>3.15</td>
<td>Safety zone of the flying aircraft</td>
<td>80</td>
</tr>
<tr>
<td>3.16</td>
<td>Example of computation of the percentage of congested sectors crossed metric</td>
<td>85</td>
</tr>
<tr>
<td>3.17</td>
<td>Flow chart of the analysis of results</td>
<td>98</td>
</tr>
<tr>
<td>4.1</td>
<td>Evolution of $\varepsilon$, the percentage of non-zero Q-Records, and the percentage of Q-Records that change 10% or more from their previous values</td>
<td>103</td>
</tr>
<tr>
<td>4.2</td>
<td>Total fuel burn when the AOCs have local or global, real-time, and accurate information</td>
<td>108</td>
</tr>
<tr>
<td>4.3</td>
<td>Total airborne conflicts when the AOCs have local or global, real-time, and accurate information</td>
<td>109</td>
</tr>
</tbody>
</table>
4.4 Total departure delay when the AOCs have local or global, real-time, and accurate information .................................................. 111
4.5 Total arrival delay when the AOCs have local or global, real-time, and accurate information .................................................. 112
4.6 %OL when the AOCs have local or global, real-time, and accurate information 113
4.7 %Osch when the AOCs have local or global, real-time, and accurate information 114
4.8 Total fuel burn when the AOCs have delayed or real-time global accurate information .................................................. 120
4.9 Total airborne conflicts when the AOCs have delayed or real-time global accurate information .................................................. 121
4.10 Total departure delay when the AOCs have delayed or real-time global accurate information .................................................. 122
4.11 Total arrival delay when the AOCs have delayed or real-time global accurate information .................................................. 124
4.12 %OL when the AOCs have delayed or real-time global accurate information .................................................. 125
4.13 %Osch when the AOCs have delayed or real-time global accurate information .................................................. 126
4.14 Total fuel burn when the AOCs have delayed or real-time local accurate information .................................................. 130
4.15 Total airborne conflicts when the AOCs have delayed or real-time local accurate information .................................................. 131
4.16 Total departure delay when the AOCs have delayed or real-time local accurate information .................................................. 133
4.17 Total arrival delay when the AOCs have delayed or real-time local accurate information .................................................. 134
4.18 %OL when the AOCs have delayed or real-time local accurate information .................................................. 135
4.19 %Osch when the AOCs have delayed or real-time local accurate information .................................................. 136
4.20 Total fuel burn when the AOCs have accurate or noisy global real-time information .................................................. 140
4.21 Total airborne conflicts when the AOCs have accurate or noisy global real-time information .................................................. 141
4.22 Total departure delay when the AOCs have accurate or noisy global real-time information .................................................. 142
4.23 Total arrival delay when the AOCs have accurate or noisy global real-time information .................................................. 144
4.24 %OL when the AOCs have accurate or noisy global real-time information .................................................. 145
4.25 %Osch when the AOCs have accurate or noisy global real-time information .................................................. 146
4.26 Total fuel burn when the AOCs have accurate or noisy local real-time information ................................................. 149
4.27 Total airborne conflicts when the AOCs have accurate or noisy local real-time information ................................................. 150
4.28 Total departure delay when the AOCs have accurate or noisy local real-time information ................................................. 151
4.29 Total arrival delay when the AOCs have accurate or noisy local real-time information ................................................. 153
4.30 %OL when the AOCs have accurate or noisy local real-time information ......................................................... 154
4.31 %Osch when the AOCs have accurate or noisy local real-time information ......................................................... 155
5.1 Contribution of each metric to the total benefits during the day for the LAX-SEA O/D pair ......................................................... 167
5.2 Contribution of each metrics to the total benefits during the day for the LAX-SEA O/D pair when AOCs have local, real-time, accurate data ......................................................... 169
5.3 Histogram for difference of route distance with respect to corresponding GCD route ......................................................... 172
List of Abbreviations

%OL Percentage of Time with at Least One Overloaded Sector. 105
%Osch Percentage of Time with at Least One Over-Scheduled Airport. 105

ABM Agent-Based Model. 34
ABMS Agent-Based Models and Simulations. 8, 34
ABS Agent-Based Simulation. 12, 34, 169
ACES Airspace Concept Evaluation System. 0, 169
AOC Airline Operations Center. 3, 9, 12, 46, 90
API Application Program Interface. 49
ARTCC Air Route Traffic Control Center. 2
ATC Air Traffic Control. 3, 4, 15
ATM Air Traffic Management. 2
ATSCCC Air Traffic System Command and Control Center. 2

CAS Complex Adaptive System. 13
CDM Collaborative Decision-Making. 4, 20, 23, 25
CTOP Collaborative Trajectory Options Program. 5, 25
DBMS Data Base Management System. 48
DEA Data Envelopment Analysis. 44

EVO Equivalent Visual Operations. 17
FAA Federal Aviation Agency. 2, 5, 13, 28
FACET Future Air Traffic Management Concepts Evaluation Tool. xix, 0, 9, 77, 169
FCA Flow-Constrained Area. 26
GCD Great Circle Distance. 3, 14, 48, 73, 78, 84
GDP Ground Delay Program. 5, 25, 26, 78
GSP  Ground Stop Program. 26

IATA  International Air Transport Association. 72

ICAO  International Civil Aviation Organization. 72

IFR  Instrument Flying Rules. 15

IMC  Instrument Meteorological Conditions. 15

IT  Information Technology. 20

JDBC  Java Database Connectivity. 48

JPDO  Joint Planning and Development Office. 17, 24

MAP  Monitor Alert Parameter. 96

MAS  Multi-Agent Simulation or Simulator. 34, 48

MASON  Multi-agent Simulation Tool. 12, 169

NAS  National Airspace System. xviii, 1, 13, 20

NASA  National Aeronautics and Space Administration. 9, 13

NATS  National Air Transportation System. 1, 13

NAVAID  Navigation Aid. 3, 14

NCF  Net-Centric Force. 4, 19

NCO  Net-Centric Operations. 4, 17, 19–21, 23–25

NCW  Net-Centric Warfare. 4, 19

NextGen  Next Generation Air Transportation System. xviii, 16

ODBC  Open Database Connectivity. 49

OEP  Operational Evolution Plan. 8

OI  Operational Initiative. 6

OMB  Office of Management and Budget. 6, 13, 28

PNT  Position, Navigation, and Timing. 17

RL  Reinforcement Learning. 1, 6, 32, 39

SESAR  Single European Air Traffic Management Research Programme. xviii

SSA  Shared Situation Awareness. 17
SUA Special User Airspace. 78
SWIM System-wide Information Management. xix, 5, 9, 10, 12, 20
TBO Trajectory-Based Operations. 17
TFM Traffic Flow Management. 2
TOS Trajectory Option Set. 26
TRACON Terminal Radar Approach Control. 2, 15
TSA Transportation Security Agency. 13
UML Unified Modeling Language. 45
UTC Universal Time Coordinated. 54
VFR Visual Flying Rules. 15
VMC Visual Meteorological Conditions. 15
Abstract

AN ANALYSIS OF THE EFFECTS OF NET-CENTRIC OPERATIONS USING MULTI-AGENT ADAPTIVE BEHAVIOR
Guillermo Calderón-Meza, PhD
George Mason University, 2011
Dissertation Director: Dr. Lance Sherry

The National Airspace System (NAS) is a resource managed in the public good. Equity in NAS access, and use for private, commercial and government purposes is coordinated by regulations and made possible by procedures, and technology. Researchers have documented scenarios in which the introduction of new concepts-of-operations and technologies has resulted in unintended consequences, including gaming. Concerns over unintended consequences are a significant issue for modernization initiatives and have historically been a roadblock for innovation and productivity improvement in the NAS.

To support the development and evaluation of the Next Generation Air Transportation System (NextGen) and the Single European Air Traffic Management Research Programme (SESAR) concepts-of-operations and technologies, analysis methodologies and simulation infrastructure are required to evaluate the feasibility and estimate the benefits. State-of-the-art NAS-wide simulations, capable of modeling 60,000 flights per day, do not include decision-making. A few recent studies have added algorithms to these simulations to perform decision-making based on static rules that yield deterministic outcomes.
In the real-world NAS, however, autonomous agents (e.g. airlines, air traffic control) are continuously adapting their decision-making strategies to achieve their enterprise objectives (i.e., minimize costs of operations). Further, analysis of an inventory of “gaming” scenarios in the NAS identified “adaptation” by agents as the underlying mechanism for taking advantage of opportunities to increase productivity in the NAS and unintended consequences. This dissertation describes: (1) the design, implementation, and integration of adaptive agent behavior in NAS-wide simulations, and (2) the use of quantitative methods to analyze the effects of adaptive behavior on the benefits of new concepts-of-operations and technology, and unintended consequences. The application of this approach is demonstrated in a case study evaluation of adaptive flightplan route selection and System-wide Information Management (SWIM) technologies using NASAs Future Air Traffic Management Concepts Evaluation Tool (FACET). The simulation results for 60,000 flights per day for more than 80 days can be summarized as follows:

1. Adaptation in flightplan route selection in the presence of SWIM resulted in a “steady-state” of the NAS that was not generated through collusion, but through self organization.

2. The steady-state in the flightplan route selection was achieved within 17 simulated days for a 60,000 flight per day NAS when global (i.e. airlines have access to data from other airlines and their own data), accurate, and real-time (i.e. no communication delay) SWIM information was available. Steady-state was achieved in 32 simulated days when the information was local (i.e. airlines have access only to their own data), real-time, and inaccurate (i.e. noisy).

3. The steady-state yielded a system-wide reduction in fuel burn (i.e. distance), departure delays, arrival delays, and airborne conflicts compared to the random selection of routes.
4. When SWIM provided global information instead of local, there was no significant effect on overall NAS performance (i.e. changes were marginal). The steady-state was reached in one additional day. Total number of airborne conflicts experienced a decrease of 2.8%, but the variability of number of conflicts was 270% higher. The variability of the total arrival delay decreased 38%, but the variability of fuel burn, departure delay, sector congestion, and arrival airport congestion did not change significantly.

5. With one day of latency in SWIM data steady-state was reached in 4 additional days with global data and 8 additional days with local data. Fuel burn did not change significantly. The total arrival delay increased 0.3% and the total departure delay increased 2.0% with global data. The total arrival delay increased 0.1%, the total airborne conflicts increased 0.7%, and the total departure delay increased 0.5% with local data. The variance decreased with global information. With local information, variance only decreased for the delays, but increased or was equal for the other metrics.

6. Inaccuracy of +/-30% in the SWIM data decreased 3.7% (2,247) the airborne conflicts with global data, and 0.9% (583) with local data. The arrival delay decreased 1.0% with global data and 1.3% with local data. The departure delay and the %OL decreased marginally too. The fuel burn increased about 0.12% (410,362 to 506,895 kg/day). The variance of the airborne conflicts increased 394%, and the arrival delay increased 103% with global data, but the variance of the departure delay and of %OL decreased 72% and 59%. With local data the variance for the total airborne conflicts increased 79%, for fuel burn increased 71%, and for arrival delay increased 51%.
The benefits of this research are: (1) the establishment of architecture and algorithms for the analysis of adaptive behavior in NAS-wide simulations (such as FACET and Airspace Concept Evaluation System (ACES)), (2) methodology for analysis of the results of adaptive behaviors in the NAS, and (3) analysis robustness to degradation of SWIM functionality of adaptive flightplan route selection. This provides the capability for researchers, analysts, and policy-makers to evaluate proposed concepts-of-operations and technologies in the presence of adaptive behavior.
Chapter 1: Introduction

System-wide simulations for the National Airspace System (NAS) are used to evaluate concepts-of-operations, technologies, and their cost/benefits. However, the state-of-the-art NAS-wide simulators lack flexibility in the behavior of the stakeholders they model. The users of these simulators must encode the decision-making in fixed sets of rules before the executions of the simulations. None of the simulators provides adaptability for the modeled stakeholders.

This dissertation investigates the inclusion of Adaptive Decision-Making behavior for airlines flightplan route selection in NAS-wide simulation tools. The adaptation is achieved by the application of Reinforcement Learning (RL) and a modification of the concept of domination to rank the performance of flights (which are multi-valued quantities).

The dissertation includes a case-study of the effects on the NAS of adaptable route selection behavior of airlines in the presence of perfect and degraded historic performance information. The results show the effectiveness of adaptive route selection in improving NAS-wide performance even when all agents use it simultaneously. This effectiveness results in benefits for multiple stakeholders and for the whole system.

This introduction summarizes the context, methodology, results, and conclusions of the research. It also presents the problem statement, contributions, and benefits of the dissertation.

1.1 The National Air Airspace System

The National Air Transportation System (NATS) of the United States is an aviation system in which public and private stakeholders interact in a complex manner to provide safe and timely domestic and international air transportation [Donohue et al., 2008]. The users of
the system are the passengers. The service providers compete for the limited resources of
the system to accomplish their individual, and often conflicting, goals.

The National Airspace System (NAS) is a complex system that is a part of the NATS and
includes all the facilities, technology, procedures, rules, and stakeholders related to the
use of the airspace of the United States [Donohue et al., 2008].

1.1.1 Structure of the NAS

The users of the NAS must follow rules and procedures to assume safety. The government
(e.g., the Federal Aviation Agency (FAA)), often in cooperation with the stakeholders of
the NAS, establishes these rules and procedures. The rules are concerned with management
of the daily operations of the NAS that can reach as high as 60,000 flights per day.

On the top of the hierarchy of the Air Traffic Management (ATM) is the Air Traffic
System Command and Control Center (ATSCCC). The ATSCCC performs Traffic Flow
Management (TFM) functions.

The continental US territory is divided into 20 geographical regions called centers. En-
route flights in a center are managed by its Air Route Traffic Control Center (ARTCC). The
Terminal Radar Approach Control (TRACON) are specialized in managing airport arrivals
and departures. TRACONs are located around airports.

Centers and TRACONs are further divided into smaller regions called sectors. A sector
or group of adjacent sectors is assigned to one controller who maintains safety inside the
sector, e.g., maintain safe separation between aircraft, and re-routing flights if bad weather
disrupts their routes. When a flight leaves a sector to enter another one, a handover pro-
cedure takes place between controllers. The handover procedure is an oral communication
between the controllers, but it could also require actions from the pilot like changes in the
codes that identify the aircraft on the radar of the controllers, possible changes in direction
or altitude. There is an upper bound in the number of flights a controller can effectively
and safely manage simultaneously, which results in an upper bound for the throughput of
a sector. The level of cognitive workload of the controllers has been used as a performance
metric for the NAS [Endsley et al., 1997, Galster et al., 2001, Majumdar, 2003, Oprins et al., 2009, Pechoucek et al., 2006].

Thousands of artificial reference points, each called a Navigation Aid (NAVAID), have been built to keep pilots from becoming disoriented when flying. The NAVAIDs are facilities that emit radios signals that, depending on the type of NAVAID, transmit different data to assist pilots in position fixing and navigation. To maintain order in the airspace, the NAVAIDs are connected by imaginary Great Circle Distance (GCD) arcs, called airways, that have names associated with them. A sequence of NAVAIDs or airway sections is called a route.

Flights follow routes from origin to destination. Figure 1.1 shows five alternate routes between Los Angeles (LAX) and New York (JFK). The labels next to the vertices of the routes are the names of the navigation aids that define the routes.

The Airline Operations Center (AOC) for each airline selects the routes in a strategic planning phase before the flights depart. The selection is reflected in a document called the flightplan that is filed with Air Traffic Control (ATC) up to 1 hour before the scheduled departure of the flight. In general, flights must adhere to the route filed in their flightplan, but modifications are allowed when the flight is already airborne.
1.1.2 NAS Performance

Measuring NAS performance is a challenging task due to its size and complexity. Metrics like total fuel burned, total delay (e.g.: departure, arrival, airborne, passenger delay, flight delay), congestion at the airports, congestion on the air (i.e. sector congestion), workload of the air traffic controllers, number of passengers, and emissions are found in reports (see files at http://catsr.ite.gmu.edu/pubs.html), web sites (see http://www.bts.gov/), and conferences to measure the performance of the NAS.

The metrics used in this dissertation are fuel burn, departure delay, arrival delay, number of airborne conflicts (as a proxy for ATC workload), percentage of time with overloaded sectors (as a proxy for airspace congestion), and percentage of time with over-scheduled arrival airports.

1.2 Next Generation Air Transportation System, NextGen

NextGen [JPDO, 2007] is a set of initiatives to modernize the NAS and allow it to accommodate the future demand. NextGen has evolved through time and taken ideas from other fields. For instance, the idea of improving synchronization between stakeholders of the system by sharing information evolved simultaneously in the aviation system with the idea of Collaborative Decision-Making (CDM), and in the military with ubiquitous flow of information [Alberts, 1996a] and the Net-Centric Warfare (NCW) [Cebrowski and Garstka, 1998] that was later refined and renamed to Net-Centric Force (NCF) [Office of Force Transformation, 2003] and Net-Centric Operations (NCO). NCO has been successfully applied in the military and in industrial settings\(^1\).

NCO emphasizes the sharing of information in a secure way: provide the right information to the right people at the right time. NCO decentralize decision making processes [Ball et al., 2001] by allowing the stakeholders to interact with each other more directly

\(^1\)See https://www.ncoic.org/home
and achieve self-synchronization [Hill et al., 2005, Jonker et al., 2007, Martin et al., 2001, Pe-
choucek et al., 2006, Alberts, 1996b, Cebrowski and Garstka, 1998] more agility and flexibility
(adaptability) [Cebrowski and Garstka, 1998], and maximum productivity, efficiency, and
usability of the public resources, while maintaining safety at the same level [Aviation IPT,
2008, NIIO, 2009]. More specific benefits like higher capacity [Aviation IPT, 2008], sustain-
ability, shared situation awareness [Ball et al., 2001], possibility of faster control [Cebrowski
and Garstka, 1998], and better support for physical distribution of the stakeholders are also
expected consequences of NextGen.

The System-wide Information Management (SWIM) is the enabling technology for
NextGen (see Net-Centric Infrastructure Services [JPDO, 2007]) that provides the means
to share information among the NAS stakeholders. It is a heterogeneous network, database,
and data collection infrastructure.

Collaborative Trajectory Options Program (CTOP)\(^2\) builds on top of CDM and the
Ground Delay Program (GDP) that has been successfully used for several years. CTOP is
a procedure to centralize the decision making about pre-departure trajectory selection in
the presence of restrictions in the NAS. The collaboration in its name implies that airlines
submit their preference\(^3\) values for trajectories to the FAA. The FAA considers the prefer-
ces of all flights affected by the restriction, and communicates the decision to the airlines.
Airlines can submit new preference values, and the FAA will revise the previous decisions
and modify the decisions to agree with the new preferences. CTOP requires a communica-
tion infrastructure, a procedure to send preferences and receive decision information, and
a centralized algorithm to make the decisions. In the first implementation steps, the same
communication means used by GDP will be used for CTOP; only some modifications in the
current messages structure are necessary to accommodate the CTOP information.

The introduction of NextGen to the NAS could bring unintended consequences [Alberts,

\(^2\)See http://cdm.fly.faa.gov/ad/CDM-GDPspecs.htm
\(^3\)In this context, the selection of action a decision-making agent would make if given the opportunity. In
particular it is the choice of flightplan route of the AOC. The preference is usually expressed by a numerical
value. The higher the value, the more preferred the action is over alternative actions, i.e. routes.
There is a robustness problem with NCO since it makes the organizations rely on communications systems, centralized data repositories, and ubiquitous data collection systems [Silbaugh, 2005]. The failure of any of them will mean missing, outdated, or false information to make decisions. Security becomes a major concern in systems that implement NCO [Alberts, 1996b] since information will be distributed among many members of the system and it will be transmitted through the communication system to different locations. There can be an information overflow in any system implementing NCO. The users of the information could be saturated with “superfluous data”. Some organizational changes are needed to successfully implement NCOs [Alberts, 1996a]. The relationships between the members of the systems will be affected and will need to adjust.

1.3 NAS-Wide Simulators and Modernization Benefits Analysis

A modernization proposal included in NextGen is called an Operational Initiative (OI). The Office of Management and Budget (OMB) is a government agency that authorizes investment for all OIs to determine their \(^4\), and to assign budget for them. OMB uses NAS-wide simulations to evaluate concepts and technologies for investment. Table 1.1 lists NAS-wide simulators in use. These tools are based on different theoretical foundations (e.g., physical laws, queuing theory, and probabilities), offer different levels of granularity (e.g., from high level aggregated metrics to flight-by-flight modeling), and fidelity (i.e., from statistical quantities to flight-by-flight deterministic quantities). The first three of these tools have been used by FAA and the OMB for NAS-wide evaluations of policy changes and investment in the NAS.

\(^4\)Value understood as a benefit with an associated cost.
Table 1.1: NAS-wide simulation tools, decision-making and adaptation functionality

<table>
<thead>
<tr>
<th>NAS-wide simulator</th>
<th>Decision-making</th>
<th>Adaptation</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>SystemwideModeler (CAASD)</td>
<td>Determined by inputs</td>
<td>Not allowed</td>
<td><a href="http://catsr.ite.gmu.edu/NASWideModelWorkshop2.htm">http://catsr.ite.gmu.edu/NASWideModelWorkshop2.htm</a></td>
</tr>
<tr>
<td>LMINET (LMI)</td>
<td>Determined by input</td>
<td>Not allowed</td>
<td>[Long et al., 1999b, Long et al., 1999a]</td>
</tr>
<tr>
<td>Future ATC Concept Evaluation Tool (FACET, NASA)</td>
<td>Via external modules</td>
<td>Possible via the API, but not included</td>
<td>[Billimoria et al., 2001]</td>
</tr>
<tr>
<td>Reorganized ATC Mathematical Simulator (RAMS, ISA Software)</td>
<td>Not explicitly allowed</td>
<td>Not allowed</td>
<td><a href="http://www.ramsplus.com/">http://www.ramsplus.com/</a></td>
</tr>
<tr>
<td>Airspace Concept Evaluation System (ACES, NASA)</td>
<td>Via external modules</td>
<td>Possible, but not included</td>
<td>[Sweet et al., 2002]</td>
</tr>
<tr>
<td>Probabilistic NAS Platform (PNP, Sensis)</td>
<td>Via external clients</td>
<td>Possible, but not included</td>
<td>[Ramamoorthy et al., 2006]</td>
</tr>
<tr>
<td>Collaborative Human-in-the-Loop Laboratory (Chill/Sim-C, ISA Software)</td>
<td>Via external modules</td>
<td>Possible, but not included</td>
<td><a href="http://catsr.ite.gmu.edu/NASWideModelWorkshop2.htm">http://catsr.ite.gmu.edu/NASWideModelWorkshop2.htm</a></td>
</tr>
</tbody>
</table>

Some of these tools allow the possibility of including decision-making in the modeled stakeholders (i.e., airlines, controllers, flights, airports) as evidenced by several studies[Sridhar et al., 2002, Wojcik, 2004, Hogan and Wojcik, 2004, Hill et al., 2005, Pechoucek et al., 2006, Wanke and Greenbaum, 2007]. However, the stakeholders modeled in these studies exhibit fixed behavior that is implied by the inputs or pre-defined by the researcher. The tools allow the inclusion of adaptive decision-making as evidenced by a study in which RL is used to implement adaptation for flight tracking devices to optimize traffic flow in a small group of sectors[Agogino and Tumer, 2008].
1.4 Problem Statement

Existing NAS-wide simulation tools used for benefit evaluations (also known as value studies\textsuperscript{5}) for concepts-of-operations and technologies only model static, predefined decision-making for the stakeholders. As a result, the studies made with the tools only model static predefined decision-making for their stakeholders\cite{Magill2001,Sridhar2002,Wojcik2004,Hogan2004,Hill2005,Pechoucek2006,Wanke2007}. However, current operations in the NAS exhibit a high degree of adaptation by all of its stakeholders. Furthermore, the introduction of NextGen will likely create growing opportunities for adaptation.

1.5 Objective

The objective of this dissertation is to solve the lack of adaptability in the NAS-simulation tools by accomplishing the following three specific objectives:

1. Design and develop the architecture and algorithms to include adaptive decision-making for the agents in a NAS-wide simulation.

2. Integrate the agents adaptive decision-making in a NAS-wide simulator.


A case study is used to demonstrate the achievement of the three objectives. The case study is an evaluation of the impact of SWIM functionality on the NAS performance in the presence of adaptive airline flightplan route selection. The evaluation is delimited by the following hypotheses:

1. The availability of system-wide information used for adaptive pre-departure flightplan route selection results in improved NAS performance and in reduced variation in the performance metrics.

\textsuperscript{5}In Economics, modern term to express the relation between a benefit and its associated cost.
2. *Latency*\(^6\) in the communication of information used for adaptive pre-departure flight-plan route selection results in decreased NAS performance and in increased variation in the performance metrics.

3. Inaccuracies in the information used for adaptive pre-departure flightplan route selection result in decreased NAS performance and in increased variation in the performance metrics.

### 1.6 Research Approach

The approach to this research is described by the following 5 steps:

1. Build an Agent-Based Models and Simulations (ABMS) of the NAS that contains the locations and capacities of, at least, the 35 busiest US airports (also known as Operational Evolution Plan (OEP)-35 airports), because these airports account for about 80% of the passengers. The model should also represent the physics of aircraft as they use the NAS. The outputs of the simulation should include the following data: fuel burn, departure and arrival delays, destination airport, current sector of each flight, and number of airborne conflicts, because these data are needed to implement adaptation.

2. Build a set of realistic input data representing 24 hours of demand for NAS. The data must include domestic, international, and general aviation (GA) flights that enter, exit, or crosses the US airspace. The input must include at least the following data: flight number, aircraft type, date and time of first appearance in the simulation, origin and destination airports (or navigation aids if they do not start at an airport), coordinates where the flight enters the simulation, cruise altitude and speed, initial heading, and flightplan.

3. Use a machine learning technique to introduce adaptation in the agent behavior.

\(^6\)A delay in the communication of data.
4. Determine the performance of the NAS in the presence of AOC’s adaptive decision-making.

5. Analyze the outputs of the simulation when AOCs adapt their route selection preferences, and SWIM provides global all-airline data, when there are delays in provision of SWIM data, and when the SWIM data are corrupted.

1.7 Summary of Results

This research integrated adaptive decision-making in a NAS-wide simulation, developed a quantitative method to analyze the results and identify performance improvements and unintended consequences. An example case-study demonstrates the application of adaptive decision-making to flightplan route selection in Future Air Traffic Management Concepts Evaluation Tool (FACET)\textsuperscript{7} to evaluate performance of the NAS in the presence of the SWIM technologies. Simulation results for 60,000 flights per day for more than 80 days demonstrated the following:

1. AOCs start the simulations selecting routes alternatives randomly. As the AOCs gain knowledge during the simulation, they start selecting preferred routes more often. The interaction of the learning processes of the AOCs results in a steady-state of the NAS, which was not achieved by collusion or trading, but through self organization. One observable result of the steady-state is a reduction in total fuel burn (i.e. distance), total departure delays, total arrival delays, and total airborne conflicts (i.e. ATC workload) compared to purely random selection of routes.

2. The learning process of the AOCs is affected by the independent variables. When SWIM provides global, real-time, and accurate data, the steady-state is reached in 17 simulated days, but steady-state is reached in 18 days when SWIM provides local, real-time, and accurate. When the SWIM delivers data with 1 day of delay, but the

\textsuperscript{7}FACET is a NAS-wide simulation developed by National Aeronautics and Space Administration (NASA).
data are accurate and global, steady-state is reached in 21 simulated days. In contrast, it takes 26 days to reach steady-state when the data are delayed 1 day, accurate, and local. With SWIM delivering inaccurate data (maximum of 30% of random error), real-time, and global, the steady-state is reached in 23 simulated days, but it takes 32 simulated days when SWIM delivers inaccurate, real-time, and local data.

3. When SWIM provides global vs. local data performance was, in general, not significantly affected. The only significant effect was an improvement in the total airborne conflicts of 2.8% (1,713 conflicts / day). The variance changed significantly for the airborne conflicts (264% increase) and the total arrival delay (38% decrease). For the other metrics, the variance was statistically equal.

4. When SWIM provides global data with a 24 hours delay vs. global data in real-time there was a significant decrease in performance for all the metrics, except for total fuel burn, which is statistically equal. The greatest change in performance was a reduction of 2.0% (+2,384 minutes) in the departure delay. The variance was, in general, statistically equal, except for the departure delay in which the experiment with global information shows a reduction of 66% in the variance.

5. When SWIM provides local data with a 24 hours delay vs. local data in real-time there were statistically significant degradations in performance for all metrics except the fuel burn. The greater degradation in performance is 0.7% (409 conflicts) in the airborne conflicts, followed by departure delay with 0.54% (642 minutes). The variance of departure delay and arrival delay showed significant reductions of 83% and 59%. The variance of the other metrics were statistically equal.

6. When SWIM provides global inaccurate data (max. 30% random error) vs. global accurate data performance was not significantly affected, except for a degradation of 0.1% (410,362 kg/day) in fuel burn, and an improvement of 3.7% (2,247 conflicts) in the airborne conflicts. The variance of the airborne conflicts increased 394%, the
variance of the arrival delay increased 103%. The variance of the departure delay decreased 72%.

7. When SWIM provides local inaccurate data (max. 30% random error) vs. local accurate data performance was degraded for fuel burn (0.2%), but it was improved 0.9% for airborne conflicts (583 conflicts), and 1.3% for the arrival delay (7,275 minutes). The variance was statistically equal for all the metrics, except the total fuel burn (increase of 71%) and airborne conflicts (increase of 79%).

8. As suggested by a preliminary study [Calderon-Meza et al., 2009], the congestion at the destination airports (metric %Osch) is not significantly altered by the flightplan route selection.

9. Congestion in the sectors (metric %OL) remains approximately constant through time in all experiments although some of its variations are statistically significant.

1.8 Contributions

From an application perspective, this research introduces adaptive airline decision-making agents in a NAS-wide simulation to evaluate the plausible effects of rules and technology changes.

This research has successfully combined a NAS-wide simulation tool, FACET, and a Agent-Based Simulation (ABS) tool, Multi-agent Simulation Tool (MASON). These tools were developed independently, by different researchers, and for different purposes.

From the theoretical perspective, this research evaluates the effects on the concurrent learning process of multi-agents of degraded accessibility, latency, and accuracy of information.

1.9 Benefits

The benefits of the this dissertation are as follows:
• The development of the architecture and algorithm needed for integrating adaptive decision-making in NAS-wide simulations.

• The development of a method to analyze the results of NAS-wide simulations with adaptive decision-making.

• The implementation of CTOP by the FAA requires each AOC to submit their preference scores for trajectory selection in flights affected by NAS restrictions. This dissertation provides the AOCs with a tool to obtain the preference scores.

• The evaluation of the effects of latency, accuracy, and availability of information in the performance of the NAS and the airlines is a step forward in the evaluation of potential failures of SWIM proposed by NextGen.
Chapter 2: Background and Literature Review

This chapter describes the background of this dissertation and reviews the literature that supports the problem statement and the methodology.

2.1 The National Airspace System

The NATS of the United States is a large and complex aviation system in which public and private stakeholders interact in a complex manner [Donohue et al., 2008]. The stakeholders of the NATS are: 1) The Congress, 2) The FAA, 3) The Department of Commerce, 4) The NASA, 5) The Department of Defense (DoD), 6) The Transportation Security Agency (TSA), 7) The OMB, 8) The regional airport authorities, 9) The airlines and their supply chain of service providers, 10) Unions, lobbying groups, homeowners near the airports, the cities, and the regional economy. The NATS is intended to enable safe and timely domestic and international air travel.

These stakeholders influence the system in several alternate ways including through the Congress. The stakeholders compete for the limited resources of the system to accomplish their particular goals. In some cases the goals conflict. In others they are complimentary or reinforce each other.

Even though the NAS is a section of the NATS, it has the six primary layers corresponding to a Complex Adaptive System (CAS) [Donohue et al., 2008]: 1) physical layer, representing the distribution of the population throughout the country, the airports, and the cities, 2) the weather layer, adding stochasticity to the system and making predictions harder, 3) government regulatory control layer, the federal laws and regulations establishing rules for competition, safety, acquisition, unions, and others, 4) the TSA/FAA layer, TSA concerns about security, FAA, about safety, 5) airline layer, airlines decide the type and
quality of service they will provide to the users, i.e. schedules, serviced cities, frequency, load factors, type of aircraft, and so on, 6) passenger/cargo layer, the actual customers of the system.

2.1.1 Navigation Aids and Airways

In the dawn of the air transportation, pilots navigated using geographical features as reference to determine their position and heading. The use of geography became problematic as longer flights were possible and as demand for “all-weather” and “all-day” operations grew. The first solution was to use fire beacons to guide the pilots at night. Later, omni-directional radio signals replaced the fire. Afterwards, these radio signals became more sophisticated and added identification and orientation information of the beacon.

The beacons, or NAVAIDs, were connected by imaginary straight lines called airways that are used by the pilots to fly between NAVAIDs. Currently, there are two types of airways, identified by the letters “J” (Julia) and “V” (Victor) followed by a numeral, for different ranges of altitude. A route is a sequence of NAVAIDs and airways.

Due to the number of NAVAIDs and airways available, a flight can select from several different routes to avoid weather, congestion, or to reduce fuel burn and delays.

Figure 2.1 shows five alternate routes between Los Angeles (LAX) and New York (JFK). One of the routes is direct, known as a GCD route, the other routes use navigation aids. The labels next to the vertices of the routes are the names of the navigation aids that define
the routes.

The flight plan must specify the airway to take when the navigation aids define intersections of several airways. The shape of the route is determined by the location of the navigation aids and the airways. In general, flights must adhere to this structure for their flight plans.

### 2.1.2 Airspace and Flight Rules

In addition to airways and navigation aids, the airspace is divided into uncontrolled airspace, controlled airspace, and positive controlled airspace [Neufville and Odoni, 2003] as shown in Table 2.1. The uncontrolled airspace is mainly used by flights under Visual Flying Rules (VFR), which do not receive any assistance from the ATC, and must maintain separation to other aircraft in the space on their own. In the United States this type of airspace is known as Class G airspace. The controlled airspace includes classes A, B, C, D, and E. Controllers are responsible for maintaining separation between pairs of Instrument Flying Rules (IFR) flights. IFR flights most maintain separation with VFR flights, and VFR flights must maintain separation with any other flight. An exception is the Class C, in which controllers issue traffic advisories and resolve conflicts for IFR/VFR pairs of flights, and issue traffic advisories for VFR/VFR pairs. In positive controlled airspace, i.e., classes A and B, most of the flights are IFR. The Class A is for en-route flights which are only IFR. Controllers are responsible for maintaining separation between flights. The Class B airspace represents the terminal of a major airport. It is controlled by a TRACON\(^1\). Controllers are responsible for maintaining separation between the aircraft. Traditionally, flights use VFR when the visibility is good, i.e., good weather, or there are Visual Meteorological Conditions (VMC). Flights use IFR when there is limited visibility, i.e., bad weather, or Instrument Meteorological Conditions (IMC). However, the current trend is toward using IFR regardless of the weather conditions.

Even with all this structure already in place in the NAS, pilots can still experience

\(^1\)TRACONs include facilities and personnel that handle arrivals and departures to one or more closely located airports.
Table 2.1: Description of the airspace classes in the NAS

<table>
<thead>
<tr>
<th>Class</th>
<th>Type</th>
<th>Description</th>
<th>Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Positive controlled</td>
<td>Used by en-route IFR flights</td>
<td>From 24,000 to 60,000 ft mean sea level (MSL)</td>
</tr>
<tr>
<td>B</td>
<td>Positive controlled</td>
<td>Centered on a major airport. The shaper resembles an inverted wedding cake.</td>
<td>From 0 ft above around the airport to 10,000 ft</td>
</tr>
<tr>
<td>C</td>
<td>Controlled</td>
<td>Around mid-sized airports. Smaller than class B but the shape is similar</td>
<td>From 0 ft above to 10,000 ft</td>
</tr>
<tr>
<td>D</td>
<td>Controlled</td>
<td>Around small airports with control tower</td>
<td>From 0 ft above to 10,000 ft</td>
</tr>
<tr>
<td>E</td>
<td>Controlled</td>
<td>Around small non-towered airports</td>
<td>0 ft above</td>
</tr>
<tr>
<td>G</td>
<td>Uncontrolled</td>
<td>Used mainly by VFR flights. Any other volume not covered by the other classes</td>
<td>Below 1,200 ft above ground and away from busy airports</td>
</tr>
</tbody>
</table>

Incomplete situation awareness, especially in IMC. ATC, in land-based monitoring stations, direct the pilots to maintain safe separation between aircraft flying the same airway or approaching a common intersection. Controllers have radars that allow them to locate all the aircraft in their assigned area and to take actions to prevent loss of safe separation and to synchronize the traffic. The communication between controllers and pilots is done by voice over traditional analog radio channels.

The organization into types of airspaces simplifies the task of the controllers since they do not have to consider the whole 3D space an aircraft could fly through, but only the airways and the intersections. The disadvantage of the structure is that the routes can be longer than the great circle distances between origin and destination, congestion is generated, especially at the intersections, and the airspace is not utilized evenly. With increasing demand the workload for the controllers also grows to the point in which the cognitive capacities of the controllers become the limitation of capacity of the airspace.

2.2 Next Generation Air Transportation System

Next Generation Air Transportation System (NextGen) is a government plan to modernize technology and procedures of the NATS. NextGen involves many stakeholders in the public and private sectors.
The existence of NextGen was required by the “Vision-100” legislation (Public Law 108-176) signed by President Bush in 2003. The law mandated the design and deployment of an air transportation system to meet the nation’s needs in 2025 [JPDO, 2007]. The same legislation created the Joint Planning and Development Office (JPDO) to lead in the implementation of the law.

The goal of NextGen is to allow the NATS to support the increasing demand while maintaining safety and security at the current levels (or better) [JPDO, 2007]. The completion of the plan is scheduled for the year 2025.

NextGen includes modification and improvements in eight capabilities [JPDO, 2007]:

1. Network-Enabled Information Access
2. Performance-Based Operations and Services
3. Weather assimilated into Decision-Making
4. Layered, Adaptive Security
5. Position, Navigation, and Timing (PNT) Services
6. Trajectory-Based Operations (TBO)
7. Equivalent Visual Operations (EVO)

With the advances in aircraft technology and automation tools, the safe distance between flights will be reduced, the responsibility of maintaining that distance will go from the controllers to the crew, and controllers workload limits will be reduced. That way, the NAS will go from clearance-based operations to TBO, and from rules-based to performance-based. All this will increase effective capacity and efficiency of the system.

The timely, robust, efficient, and secure information sharing (i.e., NCO) is a central part of the NextGen. It will support the concept of Distributed Decision-Making based on common information and provide Shared Situation Awareness (SSA) to all the stakeholders.
Table 2.2: NextGen goals and objectives as described in the Concept of Operations for the Next Generation Air Transportation System v2.0

<table>
<thead>
<tr>
<th>Goal</th>
<th>Corresponding objectives</th>
</tr>
</thead>
</table>
| Retain U.S. Leadership in Global Aviation | Retain role as world leader in aviation  
Reduce costs of aviation  
Enable services tailored to traveler and shipper needs  
Encourage performance-based, harmonized global standards for U.S. products and services |
| Expand Capacity                | Satisfy future growth in demand and operational diversity  
Reduce transit time and increase predictability  
Minimize impact of weather and other disruptions |
| Ensure Safety                  | Maintain aviation’s record as safest mode of transportation  
Improve level of safety of U.S. air transportation system  
Increase level of safety of worldwide air transportation system |
| Protect the Environment        | Reduce noise, emissions, and fuel consumption  
Balance aviation’s environmental impacts with other societal objectives |
| Ensure Our National Defense    | Provide for common defense while minimizing civilian constraints  
Coordinate a national response to threats  
Ensure global access to civilian airspace |
| Secure the Nation              | Mitigate new and varied threats  
Ensure security efficiently serves demand  
Tailor strategies to threats, balancing costs and privacy issues  
Ensure traveler and shipper confidence in system security |

The Concept of Operations for the Next Generation Air Transportation System v2.0 [JPDO, 2007] presents a table (Table 1-1 in page 1-3) with the 6 national and international goals and the 19 objectives of the NextGen. The contents of that table are reproduced in Table 2.2.

The goal of “Expand Capacity” and the capabilities 1, 2, 5, and 6 are in direct relation to this dissertation.

2.3 Net-Centric Operations

Traditionally, tactical information in the military flows vertically from the battle-field up to the command centers and back to the battle-field. The information is collected by the troops in the battle-field and orally transmitted. As technology provided faster ways to communicate information, the speed of command increased. With modern technology the information flows faster, but still only vertically in the hierarchical organization. Different groups inside the military ignore the situation of other groups, unless some higher level
command provides them with information. The time it takes for the information to flow up and down is still long, because all decisions are made at the top of the hierarchy. This reduces the efficiency and could result in increased damages, and loss of lives.

In the military, the concept of NCO evolved from experiences in the battle-field, and from concepts like the Information Warfare. The ubiquitous flow of information to all groups of the military simultaneously, was proposed in the mid nineties in an effort to increase efficiency and minimize damage[Alberts, 1996a, Alberts, 1996b]. At about the same time, the term NCW was introduced[Cebrowski and Garstka, 1998]. Currently, the term is used as a synonym for NCO, and often, for NCF [Office of Force Transformation, 2003]. The term is used in the civilian environment to imply a way for organizations to leverage the creativity and independence of all its members through the share of information. In the military of the United Kingdom the term Network Centric Operations is often used instead of NCO or NCW.

The idea of Net-Centric Operations is rooted in changes in the dynamics of economic growth and competition within and between ecosystems\(^2\) of stakeholders (e.g. businesses, armed forces)[Cebrowski and Garstka, 1998]. The new models of economic growth are based on increasing returns on investment. Competition is based in response time to changes in the environment. A justification for the adoption of NCW is the technological progress. Improvements in communications infrastructure, increase in computing power, and the proliferation of networks connecting heterogeneous computing nodes enable NCO.

The idea of the Net-Centricity is to provide “the right information” to “the right people” at “the right time”. This idea requires profound changes in the procedures of the organization, responsibilities of its members, its technological infrastructure, and its management in general. One implication is that the information can also flow horizontally in an organization. Another implication is that the decision making processes are distributed across the whole organization instead of concentrated at the higher levels. A non-obvious

\(^2\)The word ecosystem is used in this context to refer to a system in which all the parts depend on each other. It is not related to the biological meaning of the word, but it certainly is a good analogy.
implication is that the information infrastructure most be capable enough to support the increased flow of information. In modern times, the use of automation and technology (for instance sensors and a communications networks) is a good start for the success of NCO. As the organizations grow, it is hard to imagine that they could be Net-Centric without an advanced Information Technology (IT) infrastructure. However, small and localized organizations could use the principles of information sharing and distribution of decision-making without IT infrastructure.

The implementation of NCO brings benefits to the organizations. They have been applied to the military, and the industry\(^3\) [Aviation IPT, 2008]. In the NAS the application was the CDM described by several authors[Gorman et al., 1997, Neufville and Odoni, 2003, Donohue et al., 2008]. CDM is a precursor of NextGen [JPDO, 2007], and SWIM implements the infrastructure required to share timely and accurate information among the stakeholders of the NAS.

2.3.1 Principles of NCO

Several authors describe the principles of NCO are as follows [Alberts, 1996a, Cebrowski and Garstka, 1998]:

- Organizations are ecosystems of stakeholders who can be individuals or groups.
- Stakeholders show different levels of interdependency and autonomy.
- Stakeholders could be physically distributed, localized, or any degree of distribution between those extremes.
- Information must be shared among stakeholders.
- The objective of NCO is to achieve the goals of the organization (effectiveness) in the most efficient way [Alberts, 1996a].

\(^3\)See https://www.ncoic.org/home
- The better each stakeholder performs the higher performance the organization achieves.

  For this purpose, information must be timely for the stakeholders to make the best decisions.

  The NCO principles are further delimited by a set of implementable and measurable requirements. The requirements are classified in three groups each related to a quality aspect inside the organization [Alberts, 1996a, Alberts, 1996b, Cebrowski and Garstka, 1998, Office of Force Transformation, 2003, JPDO, 2007, NIIO, 2009]:

1. Quality of information

   (a) Information must show proper content.

   (b) Information must be understandable to the intended user.

   (c) Information must be accurate.

   (d) Information must be available to the intended user.

   (e) Information must be on time.

   (f) Information must be relevant for the intended user.

2. Quality of the information infrastructure

   (a) The information infrastructure must be capable enough to receive, store, and deliver the information from (to) the intended users (producers). If the organization is using a computer network, the ideas of bandwidth and storage space become important metrics.

   (b) The information infrastructure must operate at peak performance regardless of the situation.

   (c) The information infrastructure must be extensible to support future growth of the organization.

   (d) The information infrastructure must be robust.
(e) The information infrastructure must be reliable.
(f) The information infrastructure must be fault tolerant. Perhaps some redundancy will be required.
(g) The information infrastructure must be secure.
(h) The information infrastructure must be assured.
(i) The information infrastructure must reach all the users.

3. Quality of supporting tools. There must exist adequate Decision Support Tools with access to the information and at the disposal of the intended users.

These requirements characterize a Net-Centric organizations. Analysts could determine whether the organization is Net-Centric by evaluating adequate metrics related to the requirements.

### 2.3.2 Benefits from NCO to the NAS

The FAA’s NextGen Integration and Implementation Office and the describe several areas in which the application of the NCO principles will benefit the NAS [Aviation IPT, 2008,NIIO, 2009]:

- **Capacity of the Air Transportation System** will be improved by the mitigation of current constraints. The key concept here is *flexibility* (*adaptability*), achieved through the incorporation of new *Decision Support Tools*.

- **Safety** will be maintained at the same level while capacity and agility increase. The Net-Centric Operations will change the nature of the capacity vs. safety trade-off. Some examples of technologies and procedures intended to increase capacity without decreasing safety are *Net Enabled Weather, Comprehensive, Integrated Surveillance,* and *Destination-based Routing*.

- Enables Collaborative Decision Making to improve *efficiency* of the whole system [Ball et al., 2001].

23
• By matching the right information to the right people security can also be improved with Net-Centric Operations.

• The rapid adaptation to changing environments will be a signal of increased agility. The system will adapt to different cultures, weather, economies, and emergencies.

• The increase in efficiency will result in opportunities to deal better with environmental issues, i.e. increased sustainability. Some examples of technologies and processes with potential to deal better with environmental issues are: satellite-based navigation (allowing less fuel consumption), and collaborative decision making (improving coordination to reduce noise and pollution).

• Businesses can also benefit from the favorable changes in trade-offs of the Air Transportation System like: affordability vs. capacity, financial cost-effectiveness vs. efficiency, agility, and predictability, efficiency vs. environment, capacity vs. efficiency.

• Shared situation awareness: the system is a collaborative network of networks. The users of the information become also information providers. For instance, the introduction of CDM in the Air Transportation System improved performance because of the shared situational awareness, and increases satisfaction of the stakeholders[Ball et al., 2001].

• Self-synchronization [Alberts, 1996a]: the system is characterized by the autonomy of the agents and the increased value of their initiative. The possibility of organizing the system in a bottom-up fashion [Cebrowski and Garstka, 1998] allows the rapid adaptation to the environment. In the case of the Air Transportation System, the self-synchronization has been proposed and studied [Martin et al., 2001, Hill et al., 2005, Pechoucek et al., 2006, Jonker et al., 2007]. In some cases, external algorithms or procedures are needed to guarantee the safe operation of the system: the information sharing does not mean self-synchronization, but it is one step toward it. The existence of quality decision support tools is required to allow stakeholders to decide faster and
safer.

- Physical distribution of stakeholders and data collection devices: information and decisions are decentralized. NCO mitigate the effect of physical separation by providing common information to all the stakeholders.

- Faster control of the operations is achieved by eliminating the boundaries between stakeholders [Cebrowski and Garstka, 1998]. The goal is to convert a group of uninformed stakeholders into an integral organization through information sharing.

### 2.3.3 Unintended Consequences of NCO in the NAS

This section describes unintended consequences of NCO related to this dissertation [Alberts, 1996a, Alberts, 1996b, Bailey, 2004, Silbaugh, 2005]:

- Net-Centric organizations are sensitive to availability, connectivity, latency, and bandwidth of the communications infrastructure.

- Net-centric organizations depend on the ubiquitous data collection infrastructure (sensors or humans) to accurately and timely collect the right data.

- The new and more detailed information available opens opportunities for gaming when the stakeholders have conflicting goals and compete for the resources of the system [Axelrod, 1987, Wojcik, 2004, Jonker et al., 2007].

- The excess and unrestricted distribution of information can produce information overflow, that hinders people’s ability to make timely correct decisions.

The first and second consequence justifies analysis of the effects of availability of information, its timeliness, and accuracy. The third and fourth consequences are behaviors that could appear in a simulation of an NCO organization.

The authors also refer to vulnerabilities in security [Alberts, 1996b] due to the exposition of detailed information, and to required changes in the organization [Alberts, 1996a] necessary to enable NCO. However, these consequences are not in the scope of this dissertation.
2.3.4 Information Sharing and Adaptation in the NAS and NextGen

The military and the NAS are complex organizations that must adapt to the environment (e.g. weather, regulations, economy) in which they are immersed. These organizations also have a continuous need for modernization and performance improvements. The JPDO, in its Concept of Operations for the Next Generation Air Transportation System, Version 2.0, defines the relationship between NextGen and the concept of NCO [JPDO, 2007]. Specifically, chapters 4 to 9 describe the principles and requirements of NCO that apply to the Air Transportation System.

NCO are an approach to performance improvements based on information sharing, and the promotion of collaboration between the stakeholders. An example of NCO in the NAS is the CDM [Gorman et al., 1997], a precursor of NCO, introduced to the NAS in the nineties [Donohue et al., 2008].

The benefits of CDM applied to the GDPs are evaluated in an experiment that took place in late 90s [Ball et al., 2001]. The experiment included two major airports, several major airlines, and the FAA’s corresponding offices. The ideas of information exchange, persistent interaction between stakeholders, procedural improvements, tool development, and common situational awareness were already included in this first implementation of CDM. The improvements were evident in quality and timeliness of the information, reduction of delays, and accuracy of predictions. Several tools and communications infrastructure were tested during the experiment. GDPs are procedures (i.e., algorithms) in which a central controller (i.e., the FAA) limits capacity, communicates the restriction, receives information from the stakeholders, executes an algorithm to distribute the use of the limited resource, and communicates the decision to the stakeholders. The stakeholders collaborate in the sense of providing information to the central controller and receiving the decision information from it. The decision making of the stakeholders is hidden to the controller. It is implicit in the information the stakeholder sends to the central controller. The study does not evaluate the decision making of the stakeholders, but only the result of the whole process. Stakeholders create their own private strategies to obtain the most benefit from
the process.

A similar analysis for the European market was based on interviews of airlines [Martin et al., 2001]. Several information gaps were identified as needs for a future Information Management System for the Air Transportation System. The elimination of the gaps found by the study will contribute to the common situation awareness, and better information exchange between stakeholders. This study elicits the information needs of the stakeholders. It does not evaluate the strategies.

A more recent example of a NCO initiative is the CTOP. It is a way to handle en-route congestion in which the FAA assigns pre-departure trajectories to flights considering the preferences of the airlines. The program is a temporary modification to the usual rules to manage en-route traffic. The modification can be caused by reductions in capacity due to weather or other events. CTOP is intended to co-exist with the other programs like GDP, and Ground Stop Program (GSP). As with GDP and GSP, CTOP will require a communication infrastructure in place. In the beginning, CTOP will use the same protocols and communication technology used today by GDP with some extensions to accommodate for the new information needs. All these programs will greatly benefit with the introduction of SWIM.

A CTOP starts when the FAA reduces capacity in a region of the airspace, for instance due to weather, and creating a Flow-Constrained Area (FCA). Airlines react to the establishment of the FCA by computing and sending sets of alternate routes for the flights affected by the FCA. These sets of alternate routes are called Trajectory Option Set (TOS). The TOS contains preferences and constraints the airline might have for each alternate route. The FAA receives the TOS from the airlines and uses an algorithm to assign routes. The FAA communicates its decision to the airlines involved. Airlines can send modified TOSs for consideration in a revision process, or they can execute the decisions. Figure 2.2 illustrates this process with a UML activity diagram.\footnote{Figure created based on information provided in the “CTOP Industry Day” seminar on October 13, 2010.} The strategies AOCs use to compute the TOSs are unknown to other airlines’ AOCs and to the FAA. Hence, the
CTOP can be regarded as a game in which players have multiple turns, but ignore other player’s moves. Airlines can adapt to this kind of situations by learning from past decision and from system-wide information [Calderon-Meza and Sherry, 2010b, Calderon-Meza and Sherry, 2010a, Calderon-Meza et al., 2009].

TBOs are initiatives to optimize the en-route, arrivals, and departures operations as explained in the Concept of Operations for the Next Generation Air Transportation System document [JPDO, 2007] section 2.4. The optimization is achieved by using better time and space specifications of the trajectories, technology, and procedures to reduce variation in the execution of the trajectories. The term trajectory in this context means the description of a route (i.e., a sequence of points in the 3-D space) annotated with the times the flight should reach each point. TBO improves and extends the Free Flight initiative proposed and studied in the past [Adams et al., 1996, Kerns and Hahn, 1996, Endsley et al., 1997, Gorman et al., 1997, Burdun and Parfentyev, 1999, Barnett, 2000, Galster et al., 2001, Hill et al., 2005]. In Free Flight, flights are allowed to choose their route without being required to use land-based navigation aids, predetermined airways, or established rules about altitude. Free Flight requires the use of information about other flights actual situations and intentions. The information must be distributed to the flights and the air traffic controllers in order for the system to not become chaotic with all the agents making arbitrary decisions at any time.

2.4 NAS-Wide Simulators

The OMB is the government agency that determines the value and authorizes modernization initiatives for the NAS. In particular, NextGen is a plan created by the “Vision-100” legislation (see section 2.2 on page 16). Several stakeholders of the NAS are involved in the process of implementation of the initiatives. For instance, the FAA is involved when the initiative is related to safety (e.g. changes in technology and procedures). The OMB assigns budget to the initiatives if the value is high (i.e., benefits outweigh cost) and the
Figure 2.2: Activity diagram of a CTOP program.
Table 2.3: NAS-wide simulation tools, decision-making and adaptation functionality

<table>
<thead>
<tr>
<th>NAS-wide simulator</th>
<th>Description</th>
<th>Decision-making</th>
<th>Adaptation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASPAC</td>
<td>Fast-time NAS-wide simulation tool based on Queuing Theory</td>
<td>Determined by inputs</td>
<td>Not included</td>
</tr>
<tr>
<td>Systemwide Modeler</td>
<td>Fast-time NAS-wide simulator at flight level</td>
<td>Not explicitly allowed</td>
<td>Not included</td>
</tr>
<tr>
<td>LMINET</td>
<td>Represents the NAS as network of queues</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FACET</td>
<td>Physics-based (flight-by-flight), deterministic</td>
<td>External modules</td>
<td>Possible, but not included</td>
</tr>
<tr>
<td>RAMS</td>
<td>Fast-time, discrete-event, gate-to-gate, Turn-key solution</td>
<td>Not explicitly allowed</td>
<td>Not allowed</td>
</tr>
<tr>
<td>ACES</td>
<td>Physics-based (flight-by-flight), deterministic</td>
<td>External modules</td>
<td>Possible, but not included</td>
</tr>
<tr>
<td>PNP</td>
<td>Uses probabilities to include natural uncertainty of the NAS</td>
<td>External modules</td>
<td>Possible, but not included</td>
</tr>
<tr>
<td>Chill/Sim-C</td>
<td>Agent-based. Interactive, intended for research on human factors and gaming. Allows different levels of granularity and fidelity</td>
<td>External modules</td>
<td>Possible, but not included</td>
</tr>
</tbody>
</table>

return of investment, revenues, current budget indicate that the cost of the initiative can be recovered. The OMB, or its contractors, uses NAS-wide simulators and historic or projected demand and weather data as part of the process of value determination.

The aviation community has proposed, designed, and implemented several NAS-wide simulation tools. Table 2.3 and the following paragraphs describe some of those tools and its potential to include adaptive behavior for the stakeholders. References to all these tools are available from the publications section of the website of the Center for Air Transportation System Research at George Mason University⁵.

The National Airspace System Performance Analysis Capability (NASPAC) [Millner, 1993] by FAA, is a fast-time NAS-wide simulation. The system consists of a set of tools to pre-process data, simulate, and post-process simulation outputs. The simulation core is based on Queuing Theory. It has been used for about two decades to model the NAS and obtain estimates of delays and traffic flows at a NAS-wide level [Richie and Baart, 1996, Liang and Chin, 1998]. With the current version it is not easy (or even possible) to interact with each flight during the simulation. This “closed” architecture makes the

⁵See http://catsr.ite.gmu.edu
inclusion of adaptive behavior into the system difficult without changes to the system.

The *System-wide Modeler* by CAASD, is a fast-time NAS-wide simulator with a flight-by-flight resolution. The execution time for a NAS-wide simulation is between 5 and 10 minutes. The product has been recently improved with more detailed models of runways and tactical responses to congestion.

The *LMINET* [Long et al., 1999b, Long et al., 1999a], by LMI, is NAS-wide model that links queuing models for airports with sequences of queuing models for TRACONs and ARTCCs. This type of models considers the network effects that characterize the NAS. Another advantage of LMINET is its short execution time: the simulation outcome is the solution of the analytical queuing equations. The solutions are probability distributions for performance metrics. The downside is that the resolution of the simulation does not allow for details of the operations to be studied [Long et al., 2001].

The NASA’s *Future ATC Concepts Evaluation Tool* (FACET) [Bilimoria et al., 2001] is a compact, stand-alone, NAS-wide simulation tool. The tool is easy to use and its footprint in the computing system is small. It provides a Java Application Program Interface (API) that makes the control of the simulation possible from external Java code. The tool includes databases for most of the current administrative and navigation rules of current use in the NAS, i.e., sector and center attributes, navigation aids, airports, flight plans, capacities, conflict detection. The tool is physics-based and neutral in terms of procedures, i.e., no control algorithm is implemented in the model. The tool allows the individual control of many parameters of the NAS and of each individual flight. Since there is access and control to each individual flight, this tool also offers the potential to combine it with a multi-agent simulation framework and include adaptive behavior to the flights. However, the system does not include any behavior except that flights follow the physical rules at all times.

\[\text{http://catsr.ite.gmu.edu/NASWideModelWorkshop2.htm}\]
The Isa Software’s *Reorganized ATC Mathematical Simulator* (RAMS\(^7\)) is a discrete-event gate-to-gate fast-time ATC/ATM simulation package. Since RAMS is turn-key solution to air transportation modeling it does not allow for the user to expand functionality by including external models that can affect the simulation. Examples of studies performed using RAMS are a simulation of the Rumanian Air Space by Humphreys [Humphreys, 1998] and Majundar [Majumdar, 2003] dissertation.

The NASA’s *Airspace Concept Evaluation System* (ACES) [Sweet et al., 2002] is a flexible, distributed, multi-layered NAS-wide simulation framework. Its design is intended to support a wide range of different models for the NAS to cover any aspect of the system. Its layers include infrastructure for control, coordination, communications, and data analysis. It is also multi-agent which gives the potential to include different behaviors and to observe their interactions. However, the design of the system leaves to the users the implementation of agent behaviors.

The *Probabilistic NAS Platform* (PNP), based on ProbTFM [Ramamoorthy et al., 2006], developed by Sensis Corporation, is a modular NAS-wide simulator. At the center of the system is a server with a model of NAS. This model is probabilistic in the sense that is considers the uncertainties inherent to the NAS, i.e., weather, delays, capacity forecasts, speed errors. Custom-made clients can be implemented and connected to the server to use its modeling capabilities. These connectable clients offer the potential of incorporating adaptive behaviors in to the NAS-wide simulations.

The ISA Software’s *Collaborative Human-in-the-Loop Laboratory* (Chill/Sim-C)\(^8\) is a system-wide agent-based modeling platform. It can operate based on a model or with human-in-the-loop for research that include humans (e.g., gaming, human factors). This tool includes the System Wide Information Management (SWIM) functionalities. The agents in this tool can be macroscopic or microscopic, which allows the implementation of different levels of fidelity and granularity in the simulation. The possibility of defining agents allows

\(^7\)http://www.ramsplus.com/

\(^8\)http://catsr.ite.gmu.edu/NASWideModelWorkshop2.htm
the implementation of adaptable behaviors.

### 2.5 Cost and Benefit Evaluation of NAS Improvements

It is the interest of all stakeholders that the NAS operates at peak performance, remains safe, and profitable. Researchers and stakeholders have presented three types of improvements to the NAS: 1) *Increases in the physical capacity* of the system by building new airports or expanding the existing airports. There are very high costs, arduous legal and administrative processes, and long times associated to this approach. 2) *Changes in regulations and policies.* *Ground Delay Programs* (GDP) and *Collaborative Decision Making* (CDM), have been effective in the past [Donohue et al., 2008, Gorman et al., 1997, Neufville and Odoni, 2003]. These changes face opposition, require cooperation from competing stakeholders, force cultural changes, and take time and effort. There are also legitimate safety concerns regarding changes in the policies and rules. 3) *Increases the effective capacity* by optimizing the operations [Agogino and Tumer, 2008, Ball et al., 2001, Bertsimas and Patterson, 2000, Hill et al., 2005, Ramamoorthy et al., 2006, Wanke and Greenbaum, 2007, Waslander and Tomlin, 2006, Wojcik, 2004]. With this approach, the investment is expected to be lower than with other approaches. It can also bring benefits without compromising safety. The following paragraphs describe studies found in the literature about cost/benefit analyses for modernization initiatives for the NAS.

Multiagent systems and RL have been combined with a NAS-wide simulation tool (FACET) [Agogino and Tumer, 2008]. The agents in the study are not stakeholders of the NAS, but are “artificial entities” at geographical locations that aircraft must fly through. These agents achieve adaptation by using reinforcement learning to automatically learn the best actions to take. The goal of the agents is to complement the controllers by determining the actions to reduce congestion and delay in a small set of sectors. An action can be one of the three known actions to reduce congestion in the NAS: control *separation distance via Miles in Trail* (MIT), implementation of *Ground Delay Programs* (GDP), and re-routing.
The study considers coupling between agent actions, but it does not simulate the whole NAS or the actual demand of the NAS.

FACET has been used to simulate the mitigation of congestion in a limited region of the NAS by allowing controllers to use three predefined actions: MIT, GDP, and re-routing [Sridhar et al., 2002].

Two more studies use agent-based simulations to evaluate algorithms for cooperative negotiation to detect, avoid, and solve airborne conflicts in Free Flight [Hill et al., 2005, Pechoucek et al., 2006]. These studies are “proofs of concept” and they do not attempt to model the whole NAS. The negotiation processes do not learn from past negotiations, they handle the current situation only based on fixed protocols and rules. For instance, the study by Hill et al. predefines two possible behaviors for the negotiating agents: selfish and altruistic.

A study by Campbell et al. describes a collaborative system in which agents share information with other agents and can choose from three types of predefined behaviors (No-action, Obedient, Collaborative) to schedule arrivals into an airport affected by a GDP [Campbell et al., 2001]. A more recent study uses agents that can choose from three predefined behaviors (reactive, optimizing, and aggressive) and share intended route selection information to define departure schedules from an airport [Wojcik, 2004]. In both studies, the strategies of the agents are fixed or selected from a pool of predefined strategies, i.e., there is no adaptation. The modeled situations are simplified abstractions of the NAS. They serve as “proof of concept” for the idea of applying multiagent techniques to the analysis of aviation problems.

In the topic of information sharing, two studies use position and velocity of flights and cooperative negotiation to detect and solve airborne conflicts and enable the implementation of Free Flight [Wollkind et al., 2004, Hill et al., 2005]. One more study also uses position of flights, the flightplans, and negotiation to detect and solve airborne conflicts [Pechoucek et al., 2006]. In all of these studies, the flights are modeled as autonomous agents that have the ability to share information with the other agents. The decision making of the agents is
based on fixed algorithms and on the negotiations process, i.e., if the agents face the same situation several times their decisions will be always the same.

Some experimental studies focus on European air transportation systems [Magill, 2001, Oprins et al., 2009] that are smaller than the NAS and have some different rules.

Other studies take the optimization approach [Bertsimas and Patterson, 2000, Waslander and Tomlin, 2006, Waslander et al., 2006], or use mathematical models [Barnett, 2000, Wanke and Greenbaum, 2007]. Optimization studies compute actions to take to achieve the “best” performance. Due to the complexity of the problem, the algorithms used to solve the optimization problem can handle only limited regions in space or short periods of time. Furthermore, if any of the parameters of the system changes the solution must be recalculated. This limits the application of these techniques mostly to off-line situations. Hierarchical structures, i.e., aggregation, have been proposed to cope with the complexity [Bertsimas and Patterson, 2000, Waslander et al., 2006], but results for the whole NAS are not shown in the studies. The other mathematical models make several assumptions to simplify the model and make it tractable [Barnett, 2000]. Their goal is to provide insight of the effect of changes in the NAS. The assumptions set requirements that are unachievable in the real world.

### 2.6 Theoretical Foundations

This section describes the theoretical foundations for this dissertation. It describes the techniques used to model the NAS and its stakeholders (i.e., agent-based models and simulations), the techniques used to implement agent adaptation, mathematical tools used to evaluate performance of the agents (i.e., multi-objective optimizations), and the theory to predict and understand the possible outcomes of the simulations (i.e., Games Theory).
2.6.1 Agent-Based Models and Simulations

Complex systems are hard to analyze because of the non-linearity of the interactions between their stakeholders. Often, there are no analytical models to study this type of system. Agent-Based Model (ABM) and ABS are alternatives to the analytical models for complex systems. These models are also called Multi-Agent Simulation or Simulator (MAS). MAS are tools that represent real systems in a bottom-up fashion. They start with simple agents that interact in a virtual world and seek to obtain the actual macro-behavior of the real system without explicitly programming it in the behavior of the agents.

The history of ABMS can be traced to the 1940s with the von Neumann theoretical automata, that followed a set of rules defined “inside” the machine\(^9\). First, Ulam built on the von Neumann machine idea by creating “populations” of similar machines to obtain the so-called cellular automata. Cellular automata can be classified in four types. The simpler types correspond to regular languages, but other classes are universal computing machines [Wolfram, 1984]. Later J. Conway created the Game of Life\(^{10}\) built on the concept of cellular automata. A fundamental work on the concept of multi-agent models is the Dynamic Models of Segregation presented by T. Schelling [Schelling, 1969]. More contributions to the field were made by R. Axelrod (tournament of multiple agents interacting in a Prisoner’s Dilemma game)[Axelrod, 1987], C. Reynolds (flocking models)[Reynolds, 1987], J. Holland and Miller (introduction of the term “agent”).

ABMS are based on the concept of an agent. The word agent comes from the Latin verb *agere*, which means “to do.” Agents are abstractions of real entities for the analysis of systems, e.g., stakeholders in a project, nodes of a computer network, and players in a game. Agents exhibit one or more of the following properties [Rosin, 1997]: autonomy of control, perception of their environment, persistence over time, adaptation to change, and the ability to take other’s goals. The implications of having all these properties are that

---

\(^9\)This ideas were published in the set of lectures “Theory of Self-Reproducing Automata”, completed and edited by Arthur W. Burks in 1966, after von Neumann’s death.

\(^{10}\)Appeared on the October issue of Scientific America, 1970, in Martin Gardner’s column “Mathematical Games”
agents must have structure and behavior [Ferber, 1999, Garrido, 2001].

The concept of agent is very general. An agent could be a human, a computer program, a mechanical device, or any other entity that can act on the environment. Most relevant to this dissertation are the software agents. Software agents can be grouped in four classes, which could exhibit the property of “learning.” [Russell and Norvig, 2003] 1) The simple reflex agents select their actions based on the current percept without considering any history of the percept. These are simple agents that can be implemented as a set of if-then rules. 2) The model-based reflex agents keep a model of the environment together with a set of, more complex, rules. The model helps in handling partial observability of the environment. It also allows the agent to predict the effect of its actions, and how the environment will evolve in time. 3) The goal-based agents also know what the expected outcome of their existence is. They are able to plan in advance a course of action that will take them to the goal. 4) The utility-based agents introduce the concept of utility function, which is a way to measure the quality of the behavior of the agent. In this type of function, the higher the value associated to a sequence of actions, the “better” (i.e., more efficient, safer, cheaper, etc.) the behavior is considered. All rational agents try to maximize their utility, and they must behave as if they had a utility function, even if they do not actually have one.

The four types of agents described above need a way to acquire their knowledge. For simple agents, the researcher codes the behavior and models of the agents directly into them, and adjusts the behavior using the experience gained through many executions. Learning does that automatically, which adds power to the concept of agent and generally achieves better results.

The idea of ABM is to abstract the behavior of stakeholders inside agents, and make the agents interact in a virtual environment as if they were autonomous. Frequently, the result from the interaction is unexpected, i.e., it is an emergent behavior [Castle and Crooks, 2006, Epstein and Axtell, 1996]. Multi-agent models and simulations not only show emergence, but are flexible, i.e. they can be adjusted and changed very easily in comparison with other
modeling techniques, and it is often easy to establish a mental relation between the actual system and the model.[Castle and Crooks, 2006]

ABMs and ABS can be explanatory (exploratory), or predictive [Castle and Crooks, 2006]. Explanatory simulations are useful to understand theories or to formulate hypotheses. Predictive simulations can extrapolate and are mainly used for the evaluation of policy decisions and scenario development.

One limitation of these simulations is the lack of analytical models to validate them; especially in the exploratory type. Another limitation is that the simulations usually have many parameters. Many repetitions are needed to calibrate and use the simulations. Furthermore, the simulations are sensitive to initial conditions and changes in stochastic processes. This sensitivity, as well as an excess of emergence, are often more frequent than in the real systems [Castle and Crooks, 2006].

ABMSs can be verified (inner validation), calibrated, and validated following guidelines.[Castle and Crooks, 2006] The verification consists of determining if the model behaves as expected, e.g. when compared to benchmarks. Outputs should vary as expected when parameters change, every part of the model should be tested separately (first) to verify its behavior, the model should be written in a different language and the results compared; the comparison is also a validation process. The calibration, performed by adjustments in parameters or rules, is a comparison of the model outputs to the actual system outputs under similar conditions. Here, there is the risk of over-fitting.

The validation is a more rigorous comparison of the model and the real system to determine how close the AMBS is to complete validity. Statistical analysis is used to determine if the variation in the outputs are statistically significant and not a consequence of chance, and substantive, i.e., big enough to be important. The last step is to perform sensitivity analysis of the model.

Performing verification and validation is a challenging task, especially if the system being modeled cannot be controlled to observe the results of variations in the inputs and
parameters. Theoretical tools like Game Theory provide foundations to predict the outcomes of the system. The predictions can be compared to the outcomes of the model for verification purposes. Other means of verification and validation are replications of the model using different modeling techniques or different input data.

2.6.2 Game Theory

Airlines make decisions on a daily basis in the competitive market of the air transportation. The current decision-making process can be regarded as a game in which the players have only partial information of their environment and complete information about their own performance. Airlines are profit making enterprises. Their goal is to maximize their utility or profitability, hence they are rational agents. The decisions of the airlines affect the performance of the other airlines. Game Theory, described on the book The Theory of Games and Economic Behavior (1944) by John von Neumann and Oskar Morgenstern, is a suitable tool to analyze situations with several players (agents, stakeholders) making decisions simultaneously. Three components define a game in Game Theory:

1. Player: the entities making decisions. Players are also called agents.

2. Action: what a player can choose to do in a particular situation.

3. Payoff matrix: with the utility to each player for each combination of actions for all players.

The result of the game is its outcome. An outcome is Pareto optimal if it is preferred by all players. The outcome is Pareto dominated by another outcome if all players would prefer the other outcome.

The behavior of an agent is called a strategy, i.e., a mapping between situations (states) and actions. The strategy is also called policy. If an agent always takes the same decision when it is in a particular state, it follows a pure strategy. If probabilities are involved in the processes of choosing an action, the agent follows a mixed strategy. If an agent is rational, it adopts a rational strategy, i.e., it will maximize its utility regardless of the performance
of other agents in the game. The set of strategies used by all the players of a game is a strategy profile.

A strategy can strongly dominate in a game if its corresponding outcomes are better than the outcomes of any other strategy adopted by other players in the same game. A strategy weakly dominates in a game if it is better that at least one strategy and no worse than any other. A dominant strategy dominates all others. Rational agents adopt dominant strategies. If the strategy profile consists of dominant strategies only, the profile is called dominant strategy equilibrium. In an equilibrium no player will benefit from individually changing its strategy, i.e., it is an attractive local optimum in the space of strategies. According to John Nash every game has at least one equilibrium (i.e., a Nash equilibrium), but not every game has a dominant strategy equilibrium (or even a dominant strategy). In general, there could be multiple equilibrium points for a game, and their outcomes could be Pareto optimal or not. A solution of a game is strategy profile that results in a Pareto-optimal Nash equilibrium if it exists. When there are multiple equilibria, the agent must either “guess” or communicate with other agents to choose the strategies (i.e. the game of this type is called coordinated game).

The theory becomes more sophisticated when the games are repeated, i.e., when players act several times in the same game. In this dissertation, the decisions of airlines are the choices of routes. These decisions are examples of simultaneous single move games. Therefore, the cases for repeated games are not considered here.

2.6.3 Reinforcement Learning

In an ABM or Simulation the adaptation is often achieved by the interactions of the agents. The emergent behavior, frequently observed in the ABMs, is a form of adaptation to the environment at a high level, the “social level”. Therefore, adaptation is external to the individual. In other techniques like Evolutionary Algorithms, the adaptation is achieved by repeated modification of the individuals. Adaptation is internal to the individuals. Internal and external adaptations can be related if agents are internally modified during
the simulations. The modifications of the agents focus on their behavior, and it is usually achieved through some type of machine learning [Russell and Norvig, 2003].

The *supervised learning* learns functions from examples of inputs and corresponding outcomes. The *unsupervised learning* learns patterns in the input without considering the outputs.

The more general RL learns from rewards of actions. The use of experience to learn turns Reinforcement Learning into a form of *self-supervised learning*. The Equation 2.1 is a form of the *Bellman Equation* which is the foundation of the RL. The equation defines the utility of being in a state \( s \), \( U(s) \), equals the rewards received for being at state \( s \), \( R(s) \), plus the discounted future rewards obtained by following the “optimal” policy (i.e., taking the best action possible in each state). The future rewards are computed from the utilities of being in the future estates. The discount factor \( \gamma \) is a parameter of this equation that defines how important the future rewards are with respect to the current rewards. This parameter is problem dependent. The function \( T(s,a,s') \) determines which state \( s' \) is reachable from state \( s \) by taking the action \( a \). It is a model of the environment in which the agent is.

\[
U(s) = R(s) + \gamma \max_{a} \sum_{s'} T(s,a,s')U(s')
\]  
(2.1)

From a Cybernetics [Wiener, 1948] viewpoint this learning process is a closed loop system. The agent or *learner* is the *controller* that controls all or some part of a process or *plant*. The agent has a set of goals that constitute its *operation point*, and *percept* to measure the results of its actions on the process. The *percept* is the *feedback loop* of the system. Agents using RL often have no complete model of the process or environment, and no knowledge of the rules to follow to succeed. The goal of an agent using RL is to use the feedback, in the form of rewards, to learn the rules. The set of rules is also called the *policy*.

A human making decisions in a complex system would analyze current and historic data, and the established goals to base the decisions. RL uses rewards computed from present
and previous results of its actions to modify its behavior. This similarity justifies the use of RL in this research.

The act-measure-adapt-act cycle modeled by RL allows for adaptation of the agents to a changing environment. [Russell and Norvig, 2003] Adaptation is a basic need in the agents immersed in systems as complex as the NAS.

According to the classification explained by Russell and Norvig [Russell and Norvig, 2003], the NAS is a partially observable environment. The airlines do not have full access to all the current state of the NAS. The introduction of SWIM will move toward a fully observable environment without reaching it completely, because some data will remain protected by “anti-trust” laws.

The NAS is a stochastic environment. Weather is an important factor for the stochasticity of the NAS, but there are other sources of stochasticity, for instance, problems in the supply chain that supports the aviation system, or internal problems in the airline (e.g. delays in the maintenance activities of the aircraft).

In the model used in this research the NAS is an episodic environment, because the decisions of the airlines depend only on the knowledge the airline has at that moment, and on the current state of the environment. Previous decisions are not considered to decide, except in the form of the acquired knowledge. In this research, airlines make atomic and autonomous decisions for each flight they have scheduled.

The experiments model the NAS as a semidynamic environment. A dynamic environment could change while the agents are deciding which action to select. Furthermore, the performance of the agents and the outcomes of the simulations could also change with time. However, in this research, the NAS does not change while the airlines are deciding which route to select, but the performance of the agents, and outcomes of the simulation change with time.

The NAS is a continuous environment. The position, speed, altitude, heading, fuel burn, and duration of the flights are continuous. The number of possible states of the NAS is infinite. However, the model that represents the NAS is a discrete event system, the states
that are relevant for this research are discrete and finite. The schedules contain a finite number of flights. Time is handled in the simulations as a discrete quantity: the simulation time step is 1 minute. The actions of the agents are also discrete: airlines have only a finite number of alternate routes to chose from. The percept of the agents is a combination of discrete and continuous quantities.

Finally, the NAS is a multiagent environment. All Airline Operations Centers (AOC) are acting simultaneously and autonomously on the same environment.

Q-Learning Agents

The Q-Learning is a type of active RL in which agents learn their Utility Function represented by the Q-Function. The Q-Function, $Q(s,a) \rightarrow \mathbb{R}$, relates the state, $s$, and the action, $a$, to the utility (see Equation 2.2). In Equation 2.2 the parameter $0 \leq \lambda \leq 1$ is the learning rate that determines the importance given to the current rewards with respect to the previous. $R$ is the reward obtained after taking the action $a$ at state $s$, a difference to the idea of reward defined previously, but that does not affect the convergence of the Q-Function. Each pair $s,a \rightarrow x \in Q(s,a)$ is called a Q-Record. The utility represent the value of taking the action at the state. If the Q-Function is correctly learned, the Q-Learning agent can always act optimally by finding the best action to take at a given state.

$$Q(s,a) \leftarrow (1-\lambda)Q(s,a) + \lambda(R + \gamma \max_{a'} Q(s',a'))$$ (2.2)

Q-Learning does not need to use the model of the environment. These agents based their learning on history and their knowledge evolves with time, i.e. they adapt to the changes in the environment. The real agents in the NAS (dispatchers at the AOCs, ATC, pilots, etc.) are also adaptable, so the Q-learning agent is acceptable as a model of the reality.

Some complications arise when two or more agents interact, and when the environment is stochastic. In the first case, the environment is extended by all the actions taken by the other agents. In the second case, the utility becomes a probability for the action of
being successful. In both cases, the models require many more executions to obtain the optimal values. The case of the multiple agents has the complication that any change to the Q-Functions will create a “new system” that can behave in very different ways than the previous one. There is currently no definite solution to the multi-agent learning problem.

Real agents in the NAS are also subject to these type of problems. They do not know the actions other agents are going to take. They can hardly predict the future of the NAS. This makes the Q-Learning agents suitable to approximate the behavior of the real agents.

2.6.4 Comparison of Performance Vectors

In mathematics, optimization is the process of finding the minimum or maximum of a function. There are several techniques to optimize functions, but most of them apply only to continuous, single-valued functions. Single-valued functions can be ranked by comparison using relations like “greater than” or “smaller than.” Situations in which the actions of stakeholders could have effects on several aspects of a system require the use of vectors instead of scalars to measure performance [Schaffer, 1984]. In this dissertation, the performance values are vectors. Therefore, a technique to compare the multi-objective performance of actions is needed. The methods to rank multi-objective performance values can be classified in three groups [Fonseca and Fleming, 1995]: the weighted sum, target vector, and goal attainment (see Table 2.4).

A population-based approach to ranking solution proposed for the Vector Evaluated Genetic Algorithm (VEGA) resembles the weighted sum, but the weights change according to the current distribution of the objectives in the population [Schaffer, 1984].

The non-domination is a goal attainment technique to rank the solutions that is insensitive to the scales of the objectives [Goldberg, 1989, Horn et al., 1994]. However, the actual rank is still relative to the current population. Domination is used in multi-objective optimization with evolutionary algorithms [Deb, 1999] intended to find optimal configurations, i.e., the Pareto surface, of systems.
Table 2.4: Techniques to compare multiobjective performance metrics

<table>
<thead>
<tr>
<th>Technique</th>
<th>Description</th>
<th>Advantage</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weighted sum</td>
<td>Objectives are linearly combined (added) into a single number.</td>
<td>Easy to compute and rank, i.e., simple comparisons between scalars. It reduces the problem to a simple optimization of a scalar function.</td>
<td>The weights for each objective are subjective and often inconsistent between stakeholders. It requires elicitation of the weights with the stakeholders and further approximations.</td>
</tr>
<tr>
<td>Target vector</td>
<td>Solutions are ranked based on the distance to a given goal vector.</td>
<td>It can guide a progressive search toward the optimum. The ranking is easy to do, i.e., it is done by comparing the value of distance only.</td>
<td>It requires the computation of the optimal values before the technique is used. In its basic form, it is affected by the scales of the objectives (or it needs the introduction of weights for the objectives). If some other vector is used as target, the solutions are not guaranteed to be optimal. Ranks are relative to the current set of values being compared (population).</td>
</tr>
<tr>
<td>Goal attainment</td>
<td>Each objective of the vector is compared separately to a given set of acceptable values. Solutions are ranked based on how many objectives are equal or better than the given acceptable values.</td>
<td>Insensitive to the scales of the objectives. Ranking is easy to do, i.e., only scalars are compared. Suitable to directly search for the Pareto surface.</td>
<td>It is requires many comparisons. It does not include the preference of the stakeholders. Ranks are relative to the current set of values being compared (population). Not suitable to guide a progressive search for optimal values since it provides no information about how “superior” one vectors is to the other.</td>
</tr>
</tbody>
</table>

Let \( A = \langle a_1, a_2, \ldots, a_n \rangle \) and \( B = \langle b_1, b_2, \ldots, b_n \rangle \)

\[ A \text{ dominates } B \iff a_k \geq b_k \]

and

\[ \exists m \text{ such that } a_m > b_m \]
Equation 2.3 is the standard definition of domination between two vectors \( A \) and \( B \) with \( n \) elements each. Domination is not reflexive, but it is transitive, anti-symmetric, and binary, i.e., either \( A \) dominates \( B \) or it does not.

Applying domination to a search space of many dimensions requires many executions, because finding a dominant point becomes harder. Therefore, a learning process based on domination would take a long time to progress, and that progress would not be smooth. In this dissertation modification of domination is used to smoothly guide the learning process.

The Data Envelopment Analysis (DEA) technique to compare the economic efficiency of producers was introduced between the 1950’s and 1970’s [Farrell, 1957, Charnes et al., 1978]. DEA is a form of target vector in which two vectors are compared using the ratio of their magnitudes with respect to the origin of the “search space.” For this comparison to be valid, the two vectors must have the same direction. In the case of the efficiency of the producers, the comparison is always between a vector and the “optimal vector.” The optimal vector has one end at the origin and the other end on the Pareto surface. However, any two vectors with the same direction can be compared for relative efficiency. The elements of the vectors could be weighted to compensate for differences in scales of the elements or to reflect the relevance assigned to the elements.
Chapter 3: Methodology

This chapter starts by describing the concept of a NAS-wide simulation with adaptive airline decision-making in section 3.1. The section 3.2 describes the software side of the simulation using Unified Modeling Language (UML) charts and textual descriptions. The details of the database are described in section 3.3. Section 3.4 contains an explanation of the algorithms used in this dissertation. The design of experiments is the topic of section 3.5. Finally, section 3.6 describes the method used to analyze the outcomes of the simulation.

3.1 NAS-Wide Simulations with Adaptive Airline Decision-Making

The stakeholders of the NAS use strategies in an effort to control the system and obtain the most benefit while participating in the NAS operations. The stakeholders act as a distributed adaptable controller for the NAS; they continuously observe the system to find opportunities to improve their strategies.

Cybernetics, Economics, and Control Theory have studied the idea of a control system with a closed feedback loop of information like the one described in the previous paragraph. In general, a closed loop that provides negative feedback improves the stability of the system by limiting the variation of the outcomes to a range that is smaller than without the loop. It is the basis for the adaptation of the controller, without it would be difficult to know if a particular action was effective.

Figure 3.1 is a diagram of a closed-loop control system that shows the stakeholders with adaptive behavior as controllers and the NAS-wide simulation of flights as the system to be controlled (i.e. the plant). The feedback loop contains several performance metrics.
The quality and quantity of the metrics is affected by Information Infrastructure and its performance.

In this dissertation the plant of the closed-loop system is an abstraction of the NAS, a NAS-wide simulation of flights. The feedback loop is modeled by a database and the provision of access to the data by the agents to represent SWIM. The stakeholder adaptive behaviors model the (distributed) controller of the closed-loop system. The behaviors represent the knowledge of the AOCs concerning the value of selecting routes between airports and at particular times of the day. The modification of the behaviors is done by ranking the performance of current route selections with respect to archived performance records for similar situations. The outputs of the controller are the selected routes, i.e. stakeholders decisions. The data in the feedback loop are performance metrics corresponding to flightplan selections made by the airline. The outputs of the system are system-wide performance metrics and stakeholder performance metrics.

3.1.1 A Particular Case: Pre-Departure Flightplan Route Selection

A particular application, in adaptive airline flightplan route selection, demonstrates the concept presented in the previous section.

The AOCs file a flightplan some time before the scheduled departure time of the flight (see an example for a flight from LAX to JFK in Figure 3.2). Airborne re-routing is out of the scope of this dissertation.

The flightplan routes are selected from a repository of alternatives. The repository
Figure 3.2: Examples of actual flightplans and a direct route from LAX to JFK.

Figure 3.3: Distribution of flightplan route alternatives in the repository.
contains more alternatives for long distance flights than for short distance flights (see Figure 3.3). This is because there are more ways to create a path between two distant airports than between two that are closer. For any O/D pair that appears in the simulations there is, at least, a direct or GCD route. GCD routes are shortest distance routes between O/D pairs and are not sequences of land-based navigation aids as the usual routes. GDC routes are not used often in the current NAS, because some aircraft are not equipped to follow them and because they tend to complicate the work of the air traffic controllers. The complete implementation of NextGen will enable the use of GDC routes.

### 3.2 Software Perspective

This section explains the details of the software used to introduce adaptation in a NAS-wide simulation and analyze the impact of SWIM on the performance.

The simulator consists of four components (see Figure 3.4): a Data Base Management System (DBMS), a MAS, a NAS-wide Simulator, and a Main Application (marked by the pink ellipse). The design and development tasks are concentrated on the Main Application and the interfaces to the other three components. The other three components are existing applications being integrated in the simulator.

The Main Application connects to the DBMS via Java Database Connectivity (JDBC)
or Open Database Connectivity (ODBC) as required by the programming language and the DBMS itself. This interface is an abstraction layer between the Main Application and the DBMS that provides flexibility and efficiency. Similarly, the NAS-wide simulator connects to the DBMS via JDBC/ODBC, but this interface is provided by the NAS-wide simulator.

The interface between the Main Application and the NAS-wide Simulator allows the application to control the NAS-wide Simulator and to obtain information from it. In this dissertation, the NAS-wide Simulator is FACET and the interface if the Application Program Interface (API) of FACET extended with some classes to improve robustness.

The interface between the Main Application and the MAS allows the application to control the MAS and to obtain data from the agents. In the particular case of the simulator developed for this dissertation, the MAS is MASON [Luke et al., 2004] and the interface is of the public interface of MASON.

The DBMS is MySQL version 5.1.48 (or newer) in this dissertation and its interface to the Main Application is the JDBC driver “mysql-connector” version 5.1.13 (or newer).

### 3.2.1 Static View of the Simulator

The Main Application component contains 6 packages (see Figure 3.5): `facet.experiments`, `facet.experiments.parsers`, `utils`, `utils.distributions`, `utils.multiobjective`, and `utils.databases`. These packages include classes related to the interfaces shown in Figure 3.6 (e.g. the `utils.databases` package). The figure does not show the relationships with other packages. For instance, there is a relationship between `utils.databases` and the JDBC driver. There is also a relationship between the `facet.experiments` and FACET’s `facet.pkg.api_server` and `facet.pkg.api_server_api` packages. Relationships to Java’s standard libraries are also omitted in the diagram.

The package `facet.experiments` is described in the class diagram in Figure 3.6. This package is the core of the Main Application component.

The “agent” stereotype labels three classes which instances are the agents in the MAS.
(i.e., MASON): FACETAgent, Flight, and BasicAirline (or its descendant DecisionMakingAirline). These classes implement MASON’s Steppable interface. The step method of an agent implements its behavior; it is called once every time step of the simulation.

The “universe” stereotype labels the class FACETExperiment which is the shared memory in the MAS, and extends MASON’s SimState class. This class centralizes all the input and output parameters, lists of agents, and independent variables of the simulator.

The class qFunctionRecord represents a single state $\times$ action $\rightarrow$ value map in the Q-Function of an airline. The origin, destination, and time of the day compose the state. The action is the selected route. The value is the current score of the route for that state. Each airline contains many instances of this class.

The SystemMetrics class stores the values of the system performance metrics. AirlineMetrics class stores the performance metrics of a single airline. Both of these classes can save themselves in the database through the save method.

AirlinesCollection is a hash map that contains all the airline agents in a simulation. The key is the name of the airline. The class implements Java’s Iterable interface.

RouteCollection is an intermediary between the database and the application. The goal of the class is to improve performance by reducing access to the database, and to provide several multiple keys to retrieve route data.

The classes SectorUtilizationProcessor, ConfllictsProcessor, and CongestionProcessor use the raw outputs of the simulation to compute three performance metrics: a metric.
related to the sectors and the number of flights in them, the total number of airborne
conflicts, and the ratio of the actual arrival rate and the declared arrival rate of the airports.
These classes depend of lower level parsers which are classes in another package explained
later.

The classes labeled as interface are wrappers for the FACET API’s interfaces with similar
names. These classes catch occasional exceptions thrown by the FACET API interfaces and
hides them from the application.

The package face.experiments.parsers contains classes that are used by other classes in
the Main Application (see Figure 3.7). The general goal of the package is to process input
and output text files and extract information from them or transform them into a more
convenient format.

The interface Parseable introduces a layer of abstraction between the actual source
of input for the simulation and the Main Application. It provides flexibility by allowing
transparent changes in the actual source of input. The classes DBParser and PNPParser
take the input from a database and an input file for PNP (from Sensis Corporation). The
class TRXParser takes the input from a TRX (tracking) file that is one the standard files
accepted by FACET. Any of these parsers returns a sequence of FlightRecord objects, one
for each flight described in the input source. The FlightRecord objects are wrapped with
the Flight class (see above) for MASON to schedule it as an agent.

CongestionParser is used by the class CongestionProcessor (see above) to process the
arrival rates at all the airports if a text file is generated during the simulation. Congestion-
Processor can also compute the arrival rates of the airports without the parser, taking the
data directly from the FACET during the simulation.

The ConflictsParser class process the FACET-generated conflicts file and summarizes it
to count the actual number of airborne conflicts (and their duration). This class generates
an output file with the summary, but also provides the results through its public interface.

TRXFiler reads the original tracking file (TRX format) and removes the repeated ref-
ences to a flight that are typical of a tracking file. Instead, this class writes only one
Figure 3.6: UML class diagram of the facet.experiments package.
reference for each flight with the departure time (first time the flight appears in the original TRX), the target altitude, the target speed, the first coordinates of the flight (on the first time it appeared in the original TRX), and the rest of attributes for a flight that do not change in the multiple references of the original TRX (i.e., flight number, aircraft type, flightplan).

Merger is a class used when the simulation generates the outputs as text files. This class merges several output files into a single file. This is useful for the presentation of the results.

The package utils contains several classes (see Figure 3.8) that are used throughout the Main Application, but that cannot be grouped in any other package because of their broad range of goals.

The AirportCodeConverter static class\textsuperscript{1} is used to convert airport codes from their 3-character versions to their 4-character ones and vice-versa. It can also test if a code corresponds to the US airport. This class is introduced due to the “context dependent”

\textsuperscript{1}A static class in this context (the Java programming language context) is a class that cannot be instantiated.
use of airport codes observed in the input files. Airport codes of 3 and 4 characters for US airports are mixed in a single file. Most of the times 4-characters are used when the flight described is international, so at least one of its airports is in the US. For domestic flights 3-characters are almost always used. The data to determine if the flight is international or not are not explicit in the input file. Sometimes flights are not described by origin and destination airports, but by coordinates or just navigation aid names. This class is intended to filter all these cases and answer with key values that identify the case when this is possible.

The Coordinates static class converts between the different coordinate formats found in input files and used inside of FACET, and provides other functions related to coordinates. This class is used in the pre-processing of the input file and during the simulation.

The TimeMapper static class computes the minutes of the day from a time stamp value. The time stamp value is assumed to be in milliseconds from the January 1, 1970 at 00:00:00 Universal Time Coordinated (UTC). This class generalizes knowledge acquired during the simulation by abstracting out the day, month, and year information in the time stamps.

The DualKeyHashMap is a generic class\(^2\) to implement a hash table with two keys. The

\(^2\)Generic is the term in Java for a template in C++. Though there are some differences between the two concepts, from the point of view of the programmer they are the same idea.
unique key can be of any given non-simple type. The second key (not necessarily unique) is a String. This class is useful for the collection of flight agents since flights are uniquely identified by an integer number given by FACET, but it is convenient to search for them by the flight number too; the flight number is actually a string.

The `utils.databases` package contains only the DatabaseConnection class (see Figure 3.9). This class is a database connection pool. Its goal is to enable the parallel processing of the database connections by maintaining several open connections and assigning them on demand whenever they are available. This class is implemented because the later stages of the simulation require many agents writing information to the database. Observations of the use of resources by the Main Application indicated that the processors of the computer were sub-utilized during this stage, but there the application was slow because it had to wait until each database access was complete to start the next access. Then this later stages of the simulation were parallelized by the use of standard Java thread pools to exploit the multiple processors of the computer, and the connection pools was introduced to served the threads as fast as possible.

The package `utils.distributions` (see Figure 3.10) provides an exponentially distributed random number generator, and a class that always returns a constant value. These classes are used to push-back delays for the flights in the simulation. If the simulation is configured not to use push-back delays, the ConstantDistribution (actually not a distribution) is used. This is a programming resource to generalize the process of push-back delays generation. If the simulation is configured to use push-back delays the ExponentialDistribution is used. A call the next method of these classes generates the next random number with
the corresponding distribution. The parent static class ProbDistribution uses the MersenneFastTwister random number generator (uniformly-distributed) provided by the package `ec.utils` from FACET.

The package `utils.multiobjective` includes the classes to implement the goal attainment performance ranking techniques explained above (see Figure 3.11).

The PerformanceComparator interface abstracts the idea of comparing a single vector to a set of vectors, and the notion of computing its rank with respect to the set.

The class DominationComparator implements the PerformanceComparator interface. It implements the ranking using the standard domination as described in section 2.6.4.

The class pDominationComparator also implements the PerformanceComparator interface. It implements the ranking using a modified domination operator that will be described in section 3.4.4.
Table 3.1 summarizes the total number of lines of code (LOC) for all the classes in the Main Application component.

### 3.2.2 Dynamic View of the Simulator

Figure 3.12 shows a simplified UML sequence diagram of the simulator. Some messages are omitted for clarity, but the main interactions between objects are shown in this diagram. The simulation starts when the MASON framework calls the method `start` of the object `experiment` (call not shown in this diagram). The object is a FACETExperiment which descends from the MASON’s `SimState`. Instances of the FACETServerAPI and of MASON’s Schedule exist throughout the execution of the simulation. These instances are created and maintained by their corresponding execution components (see Figure 3.4 on page 48).

The only instance of FACETExperiment creates a single instance of `DatabaseConnection` (i.e., database) that is static and public. All the other objects use this instance to obtain connections to the database, but these interactions are not shown in the diagram.

There are also single instances of FACETAgent, conflicts (ConflictsProcessor), utilization (SectorUtilizationProcessor), and congestion (CongestionProcessor). However, there are multiple instances of airlines (DecisionMakingAirline) and flight (Flight).

The MASON scheduler calls the method `step` of each Steppable object it has registered once every time step of the simulation. The step methods of the registered Steppables are called in random order. However, agents can be grouped and each group can be assigned a number that defines the order in which MASON will call the step methods for each group. Inside the group, the step methods are called in random order. Currently, the groups and their order are: group 0 with FACETAgent only; group 1 with about 60,000 flights. The airlines should be group 2 with about 1,000 agents, but, as the diagram indicates, the current implementation does not use the airlines as explicit agents.

The message `stop` sent by the flights to themselves happens when the flight is on the ground after landing. In other states, the flight uses its step method to compute flight-related performance metrics like fuel burn and distance. This message actually takes the
Table 3.1: Lines of code (LOC) for all the classes in the Main Application component

<table>
<thead>
<tr>
<th>Package</th>
<th>Filename</th>
<th>Blank</th>
<th>Commented</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>facet.experiments</td>
<td>AirlineMetrics.java</td>
<td>12</td>
<td>42</td>
<td>70</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>AirlinesCollection.java</td>
<td>23</td>
<td>37</td>
<td>81</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>BasicAirline.java</td>
<td>16</td>
<td>85</td>
<td>21</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>ConflictsProcessor.java</td>
<td>16</td>
<td>74</td>
<td>120</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>CongestionProcessor.java</td>
<td>34</td>
<td>114</td>
<td>197</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>DecisionMakingAirline.java</td>
<td>70</td>
<td>252</td>
<td>465</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>FACETAgent.java</td>
<td>27</td>
<td>58</td>
<td>168</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>FACETExperiment.java</td>
<td>218</td>
<td>516</td>
<td>1197</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>Flight.java</td>
<td>77</td>
<td>242</td>
<td>274</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>InputParamsRecord.java</td>
<td>17</td>
<td>44</td>
<td>82</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>ProtectedAircraftInterface.java</td>
<td>160</td>
<td>870</td>
<td>190</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>ProtectedAirportInterface.java</td>
<td>41</td>
<td>169</td>
<td>114</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>QFuncMessageProcessor.java</td>
<td>28</td>
<td>76</td>
<td>131</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>QFunctionRecord.java</td>
<td>14</td>
<td>62</td>
<td>63</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>Route.java</td>
<td>24</td>
<td>23</td>
<td>93</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>RoutesCollection.java</td>
<td>45</td>
<td>101</td>
<td>261</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>SectorUtilizationProcessor.java</td>
<td>98</td>
<td>298</td>
<td>514</td>
</tr>
<tr>
<td>facet.experiments</td>
<td>SystemMetrics.java</td>
<td>13</td>
<td>62</td>
<td>45</td>
</tr>
<tr>
<td>facet.experiments.parsers</td>
<td>AttributesProcessor.java</td>
<td>26</td>
<td>51</td>
<td>89</td>
</tr>
<tr>
<td>facet.experiments.parsers</td>
<td>ConflictsParser.java</td>
<td>26</td>
<td>87</td>
<td>129</td>
</tr>
<tr>
<td>facet.experiments.parsers</td>
<td>CongestionParser.java</td>
<td>25</td>
<td>69</td>
<td>95</td>
</tr>
<tr>
<td>facet.experiments.parsers</td>
<td>DBParser.java</td>
<td>27</td>
<td>72</td>
<td>221</td>
</tr>
<tr>
<td>facet.experiments.parsers</td>
<td>FlightRecord.java</td>
<td>11</td>
<td>80</td>
<td>57</td>
</tr>
<tr>
<td>facet.experiments.parsers</td>
<td>KeyLngStr.java</td>
<td>8</td>
<td>34</td>
<td>41</td>
</tr>
<tr>
<td>facet.experiments.parsers</td>
<td>KeyStrLng.java</td>
<td>7</td>
<td>35</td>
<td>41</td>
</tr>
<tr>
<td>facet.experiments.parsers</td>
<td>Merger.java</td>
<td>18</td>
<td>58</td>
<td>116</td>
</tr>
<tr>
<td>facet.experiments.parsers</td>
<td>Parseable.java</td>
<td>1</td>
<td>25</td>
<td>6</td>
</tr>
<tr>
<td>facet.experiments.parsers</td>
<td>PNPParser.java</td>
<td>23</td>
<td>71</td>
<td>110</td>
</tr>
<tr>
<td>facet.experiments.parsers</td>
<td>QFunctionsMerger.java</td>
<td>34</td>
<td>118</td>
<td>116</td>
</tr>
<tr>
<td>facet.experiments.parsers</td>
<td>SectorUtilizationParser.java</td>
<td>27</td>
<td>125</td>
<td>97</td>
</tr>
<tr>
<td>facet.experiments.parsers</td>
<td>TRXFilter.java</td>
<td>41</td>
<td>122</td>
<td>277</td>
</tr>
<tr>
<td>facet.experiments.parsers</td>
<td>TRXParser.java</td>
<td>51</td>
<td>171</td>
<td>291</td>
</tr>
<tr>
<td>utils</td>
<td>AirportCodeConverter.java</td>
<td>8</td>
<td>40</td>
<td>34</td>
</tr>
<tr>
<td>utils</td>
<td>Coordinates.java</td>
<td>15</td>
<td>75</td>
<td>55</td>
</tr>
<tr>
<td>utils</td>
<td>DualKeyHashMap.java</td>
<td>17</td>
<td>12</td>
<td>75</td>
</tr>
<tr>
<td>utils</td>
<td>GenericMessage.java</td>
<td>9</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>utils</td>
<td>KeyBuilder.java</td>
<td>1</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>utils</td>
<td>TimeMapper.java</td>
<td>4</td>
<td>24</td>
<td>10</td>
</tr>
<tr>
<td>utils.databases</td>
<td>DatabaseConnection.java</td>
<td>22</td>
<td>170</td>
<td>117</td>
</tr>
<tr>
<td>utils.databases</td>
<td>DBUUpdate.java</td>
<td>28</td>
<td>41</td>
<td>150</td>
</tr>
<tr>
<td>utils.distributions</td>
<td>ConstantDistribution.java</td>
<td>8</td>
<td>38</td>
<td>20</td>
</tr>
<tr>
<td>utils.distributions</td>
<td>ExponentialDistribution.java</td>
<td>9</td>
<td>58</td>
<td>44</td>
</tr>
<tr>
<td>utils.distributions</td>
<td>ProbDistribution.java</td>
<td>8</td>
<td>37</td>
<td>12</td>
</tr>
<tr>
<td>utils.multiobjective</td>
<td>DominationComparator.java</td>
<td>6</td>
<td>20</td>
<td>29</td>
</tr>
<tr>
<td>utils.multiobjective</td>
<td>PDominationComparator.java</td>
<td>7</td>
<td>24</td>
<td>39</td>
</tr>
<tr>
<td>utils.multiobjective</td>
<td>PerformanceComparator.java</td>
<td>1</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td><strong>Total</strong></td>
<td>1401</td>
<td>4805</td>
<td>6395</td>
</tr>
</tbody>
</table>
flight agent out of the schedule. MASON handles this message and all the operations related
to the de-scheduling of the agents.

Similarly, the message *kill* from the FACETAgent to the experiment only occurs when
FACETAgent determines that the simulation has reached its time limit. MASON’s *SimState*
class, the ancestor class of the FACETExperiment, handles this message by calling the *finish*
method of the class and terminating the multi-agent simulation. The messages shown after
the finish message are the response to the finish message. Most of these messages are
actually executed concurrently. This includes the process message to the airlines (about
1,000 objects) which are executed concurrently using a standard Java thread pool and
multiple connections to the database provided by the database object.

Since FACETAgent is a reflex agent, it does not have state, but only reacts to external
stimulus. If the FACET simulation is still running the agent will call three messages in
sequence: recordFlightAgent, countSectorUtilization and step. If the FACET simulation
reached its time limit, then the agent call the method *kill* to terminate the FACET and the
MASON simulations.

The behavior of the flight agent is illustrated by the state diagram of Figure 3.13. During
the state of Flying, the agent updates its traveled distance and fuel burn. It also computes
the number of congested sectors it crosses which is a metric used in the adaptation process.

### 3.3 The Database

The simulation uses a database to store results and data that must persist between exe-
cutions. The database supports storing and retrieving the metrics for individual flightplan
route selections (i.e., flights), airlines, the system, the Q-Functions for each airline, and
information about the routes. Figure 3.14 is an entity relationship diagram of the database.
Table 3.2 is the data dictionary.

The table EXECUTIONS records the executed experiments and the experiments being
executed. The *treatment* is a unique experiment identification selected by the experimenter.
Figure 3.12: UML sequence diagram for one execution of the simulator.
Figure 3.13: UML state diagram for the Flight agent.
Figure 3.14: Design of the database for the simulator.
In the beginning of an execution\(^3\), the simulation computes the next execution number, also called simulated day, for the given treatment and creates a new record with the treatment and the execution number in this table. The table also records statistics for the experiment.

The table EXP-PARAMS records the values of the independent variables for the experiments.

The table SYSTEM\_METRICS records the values of the system-wide performance metrics for all experiments, and all executions.

The table AIRLINE\_METRICS records the values of the airline performance metrics for all airlines, all experiments, and all executions.

The table SELECTION\_METRICS records the values of the performance for individual flights regardless of whether the flights were processed by the route selection algorithm or not.

The table QFUNCTIONS stores the airlines Q-Functions. This table records only the most recent values of the Q-Records, not the history Q-Records changes. So, the table has no reference to the execution number. This is done to reduce the growth of the database. For about 60,000 operations day and about 80,000 recorded routes, there are 330,000 Q-Records. This lack of history also justifies some of the attributes in the table EXECUTIONS that record statistics about the changes in the Q-Records.

The table ROUTES stores all the route options available from which the airlines can choose. The contents of this table were taken from example files with routes used actual flights. Ideally, this table could grow indefinitely because there are many possible routes between two airports.

Table 3.2: Data dictionary of the database

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Treatment</td>
<td>int(6)</td>
<td>Foreign key (exp_params)</td>
</tr>
</tbody>
</table>

\(^3\)A synonym for simulated day. It is usually associated to the treatment (i.e., a unique experiment identifier).
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution</td>
<td>int(6)</td>
<td>The number of the simulated day during an experiment</td>
</tr>
<tr>
<td>Finished</td>
<td>bit</td>
<td>An flag to signal the completion of the experiment (0 = executing, 1 = finished)</td>
</tr>
<tr>
<td>tot_flights</td>
<td>int()</td>
<td>Total number of flights read from the input</td>
</tr>
<tr>
<td>decision_flights</td>
<td>int()</td>
<td>Number of flights processed by the route selection algorithm</td>
</tr>
<tr>
<td>tot_airlines</td>
<td>int()</td>
<td>Total number of airlines inferred from the input</td>
</tr>
<tr>
<td>duration</td>
<td>Real</td>
<td>Duration of the execution (1 day) in minutes</td>
</tr>
<tr>
<td>tot_routes</td>
<td>int()</td>
<td>Total number of distinct routes</td>
</tr>
<tr>
<td>tot_qrecs</td>
<td>int()</td>
<td>Total number of Q-Records for the execution</td>
</tr>
<tr>
<td>qrecs_non_zero</td>
<td>int()</td>
<td>Total number of Q-Records which value is greater than 0</td>
</tr>
<tr>
<td>tot_od_pairs</td>
<td>int()</td>
<td>Total number of O/D pairs represented in the Q-Records</td>
</tr>
<tr>
<td>od_pairs_non_zero</td>
<td>int()</td>
<td>Total number of O/D pairs that contain at least one Q-Record with a value greater than 0</td>
</tr>
<tr>
<td>Epsilon</td>
<td>Real</td>
<td>The $\varepsilon$ parameter of the route selection algorithm</td>
</tr>
<tr>
<td>Lambda</td>
<td>Real</td>
<td>The $\lambda$ parameter of the learning algorithm</td>
</tr>
</tbody>
</table>

Table: exp_params
<table>
<thead>
<tr>
<th><strong>Attribute</strong></th>
<th><strong>Type</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>treatment</td>
<td>int(6)</td>
<td>Id for the experiment and its replications</td>
</tr>
<tr>
<td>push_back_delay</td>
<td>Real</td>
<td>Values of the push-back delay parameter for the experiment (average delay in minutes)</td>
</tr>
<tr>
<td>speed_error</td>
<td>Real</td>
<td>Value of the speed error parameter for the experiment (one standard deviation in knots)</td>
</tr>
<tr>
<td>availability</td>
<td>Varchar(6)</td>
<td>Value of the availability parameter for the experiment (airline, all)</td>
</tr>
<tr>
<td>latency</td>
<td>int(3)</td>
<td>Value of the latency parameter (number of days)</td>
</tr>
<tr>
<td>accuracy</td>
<td>Real</td>
<td>Value of accuracy parameter (percent from the mean)</td>
</tr>
</tbody>
</table>

Table: system_metrics

<table>
<thead>
<tr>
<th><strong>Attribute</strong></th>
<th><strong>Type</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>treatment</td>
<td>int(6)</td>
<td>Foreign key (executions)</td>
</tr>
<tr>
<td>execution</td>
<td>int(6)</td>
<td>Foreign key (executions)</td>
</tr>
<tr>
<td>fuel_burn</td>
<td>Real</td>
<td>Total fuel burn (in kg)</td>
</tr>
<tr>
<td>conflicts</td>
<td>int()</td>
<td>Total number of airborne conflicts</td>
</tr>
<tr>
<td>departure_delay</td>
<td>int()</td>
<td>Total departure delay (in minutes)</td>
</tr>
<tr>
<td>arrival_delay</td>
<td>int()</td>
<td>Total arrival delay (in minutes)</td>
</tr>
<tr>
<td>percent_time_overloaded</td>
<td>Real</td>
<td>Percentage of simulated time in which at least one sector is overloaded</td>
</tr>
<tr>
<td>percent_airport_overscheduled</td>
<td>Real</td>
<td>Percentage of simulated time in which at least one destination airport is overscheduled</td>
</tr>
</tbody>
</table>
### Table: airline_metrics

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Treatment</td>
<td>int(6)</td>
<td>Foreign key (system_metrics)</td>
</tr>
<tr>
<td>Execution</td>
<td>int(6)</td>
<td>Foreign key (system_metrics)</td>
</tr>
<tr>
<td>Airline</td>
<td>Varchar(5)</td>
<td>Airline code</td>
</tr>
<tr>
<td>fuel_burn</td>
<td>Real</td>
<td>Cumulative fuel burn</td>
</tr>
<tr>
<td>Conflicts</td>
<td>int()</td>
<td>Number of conflicts a flight from the airline is involved in</td>
</tr>
<tr>
<td>departure_delay</td>
<td>int()</td>
<td>Cumulative departure delay (in minutes)</td>
</tr>
<tr>
<td>arrival_delay</td>
<td>int()</td>
<td>Cumulative arrival delay (in minutes)</td>
</tr>
<tr>
<td>Distance</td>
<td>Real</td>
<td>Cumulative distance traveled by the flight of the airline</td>
</tr>
<tr>
<td>q_records</td>
<td>int()</td>
<td>Number of Q-Records for the airline</td>
</tr>
<tr>
<td>non_zero_qrecs</td>
<td>int()</td>
<td>Number of Q-Records with value greater than 0</td>
</tr>
<tr>
<td>change_qrecs</td>
<td>int()</td>
<td>Number of Q-Records that changed more than 10</td>
</tr>
</tbody>
</table>

### Table: selection_metrics

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Treatment</td>
<td>int(6)</td>
<td>Foreign key (airline_metrics)</td>
</tr>
<tr>
<td>execution</td>
<td>int(6)</td>
<td>Foreign key (airline_metrics)</td>
</tr>
<tr>
<td>airline</td>
<td>Varchar(5)</td>
<td>Foreign key (airline_metrics)</td>
</tr>
<tr>
<td>time</td>
<td>int()</td>
<td>Scheduled departure (in minutes from the start of the day) of the flight</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Treatment</td>
<td>int(6)</td>
<td>Foreign key (exp_params)</td>
</tr>
<tr>
<td>Airline</td>
<td>Varchar(5)</td>
<td>Foreign key</td>
</tr>
<tr>
<td>Time</td>
<td>int()</td>
<td>Time of the day the selection was made (in minutes from the start of the day)</td>
</tr>
<tr>
<td>route_seq</td>
<td>int(8)</td>
<td>unsigned Foreign key (routes)</td>
</tr>
<tr>
<td>Value</td>
<td>Real</td>
<td>Value of the Q-Record. Score of the route that defines the <em>preference</em> of the airline</td>
</tr>
</tbody>
</table>

Table: Routes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>route_seq</td>
<td>int(8)</td>
<td>Unique id for the route</td>
</tr>
</tbody>
</table>
The data dictionary describes the types of the attributes in a general form (e.g., int(5)). The number of digits is specified to give an idea of the accepted range of numbers. However, the actual type used to create the tables depends on the Database Management System (DBMS) used, and often, on the hardware on which it runs. If not configured otherwise, the simulation uses MySQL, and the engine is InnoDB to utilize the relational database features offered by MySQL. The simulation can connect to any database that supports JDBC.

### 3.4 Algorithms for Main Application Component

The following sections describe the algorithms implemented by the Main Application component (see Error: Reference source not found), the formats of the inputs and outputs, and the meaning of the configuration file.

#### 3.4.1 The Flight Specifications Input

The NAS-wide simulator requires specifications of the flights that participate in the period of time being simulated. In general, the data required to specify a flight are the following:

1. Flight ID. Preferably unique for the period being simulated.

2. The date and time of first appearance in the period being simulated. If the flight is still on the ground or taking off when it appears for the first time this time can be the schedule departure time or the wheels off time. If the flight is already flying when

---

4Version 5 or better.
it appears in the simulation this time corresponds to the moment in which the flight entered the NAS like an international flight entering the NAS.

3. Aircraft type, type of engines, weight, etc. This is required to compute the fuel burn of the flight.

4. Position of the flight at the moment of the first appearance in the simulation. This could be the coordinates or any landmark from which coordinates can be obtained like an airport code.

5. Speed and altitude at the moment of first appearance in the simulation.

6. Target cruise speed and altitude. These two data are related to the aircraft type, but there are always ranges of operations for each type of aircraft.

7. Origin and destination of the flight. These could be, ideally, ICAO 4-character airport codes, but they could also be coordinates or landmarks.

8. Preferably, the scheduled arrival date and time (if any).

FACET can take input from files or from real-time streams, i.e., connecting directly to web-based data services. Several formats are valid for the input files. The most common formats are Aircraft Situation Display to Industry (ASDI) and the flight tracking data file identified by TRX. Files in ASDI and TRX are text files in which each line represents a record. The ASDI format provides 10 types of records [CSC, 2009]. ASDI files can be used in FACET to simulate flights and to play back the input data.

The TRX file allows two types of records. One type describes the time stamp of the following records. The other type describes the tracking data of each flight in a time stamp, i.e., flight ID, aircraft type, coordinates, flight level (altitude), speed, heading angle, current center and sector (only valid for the USA), filed speed, and the current flightplan string. This type of file can also be used for simulation and play back. This is the format used in this dissertation.
The syntax of the TRX file is described by the BNF-like Grammar 3.1 and the assumptions stated below.

**Grammar 3.1** Syntactic specification of the input data for the operations

```
⟨flight specs⟩→(⟨timestamp record⟩⟨track record⟩)+
⟨timestamp record⟩→TRACK_TIME⟨digit⟩{10}
⟨track record⟩→TRACK⟨flight id⟩⟨aircraft type⟩⟨latitude⟩⟨longitude⟩⟨speed⟩⟨level⟩
⟨heading⟩⟨center⟩⟨sector⟩⟨filed_speed⟩⟨fp⟩
⟨flight id⟩→⟨alpha⟩+(⟨digit⟩)
⟨aircraft type⟩→⟨alpha⟩{(⟨alpha⟩⟨digit⟩)+
⟨latitude⟩→⟨digit⟩{5-7}
⟨longitude⟩→2⟨digit⟩{5-7}
⟨speed⟩→⟨digit⟩{1-3}
⟨level⟩→⟨digit⟩{1-3}
⟨heading⟩→2⟨digit⟩{1-3}
⟨center⟩→⟨alpha⟩+
⟨sector⟩→⟨alpha⟩{(⟨alpha⟩⟨digit⟩)+
⟨filed_speed⟩→⟨digit⟩{1-3}
⟨fp⟩→FP_ROUTE⟨alpha⟩{(⟨alpha⟩⟨digit⟩)+(⟨|.||/⟩)
⟨alpha⟩{(⟨alpha⟩⟨digit⟩*)}+
⟨alpha⟩→a-zA-Z
⟨digit⟩→0-9
⟨whitespace⟩→LF|CR|SPACE|TAB
```

Where symbols between “<” and “>” are non-terminals, and symbols in bold are terminals. The topmost rule of the grammar has the start symbol in its left side. The symbols “LR”, “CR”, “SPACE”, and “TAB” are terminals that represent the characters with the same names. The “+” means “one or more occurrences of the previous token or expression”. The numbers in curly parenthesis indicate the number of repetitions allowed, i.e., \{n\} means n exactly repetitions, \{n-m\} means from n to m repetitions. The “?” superscript means that the previous token or expression is optional. Round parenthesis combine tokens or expressions in groups.

The following conditions apply for the tokens of the Grammar 3.1.

---

5BNF: Bakus Naur Form is a formal grammar specification.
• The flight ID is the flight number known to the passengers of the flights.

• The aircraft type is an ID, such as A320 and B747, that allows the simulation to determine the performance profile of the aircraft. This information determines the maximum and optimal speed and altitude, and the fuel burn rates of the aircraft. A generic type of aircraft, used when there is no information about the type is ALOR2.

• The TIMESTAMP_RECORD is the number of seconds from January 1, 1970 00:00:00 UTC. This time is the same for all the following records until another TIMES-TAMP_RECORD changes the time or until the end of the file is reached.

• A negative sign means “south” for the latitude and “west” for the longitude. The two right-most digits of the LATITUDE and LONGITUDE fields are the number of seconds (00 to 59), the next two digits to the right are the number of minutes (00 to 59), and the remaining digits (1, 2 or 3 digits) are the number of degrees.

• Speeds are positive integer numbers in nautical miles per hour (i.e., knots). The two speed fields cannot be 0 at the same time. The “filed speed” is the target speed. The “speed” is the actual speed.

• The levels are integer numbers in hundreds of feet. This level is the target level of the flight.

• The headings are integer numbers in degrees measured clockwise. North is 0 degrees, west is -90 degrees, south can be 180 or -180, and east is 90 degrees. Also, greater integer numbers are valid, e.g. 270 means west.

• Center and sector names are only valid for flights currently in the United States airspace. If the flight is flying over the oceans or if there is no information, the value ‘NONE’ can be used for these fields.

• The flightplan is a string representing the sequence of navigation aids (including the origin and destination airports if any) the flight will follow during the simulation. The simplest flightplan is a direct route represented by ORIGIN..DESTINATION.
• The airport codes are 4-character International Civil Aviation Organization (ICAO) for international flights, and 3-character International Air Transport Association (IATA) for purely domestic flights.

Since input data could come from different sources, an abstraction layer in the form a parser (TRXParser class), is implemented in the Main Application component. The function of the parser is to hide the details of the actual input source and provide simulator with a standard source of input data.

This process modifies the original TRX file with the route selections made by the airlines before takeoff and, optionally, with push back delays and speed errors. The process is controlled by the FACETExperiment class making use of almost all the other classes, but especially TRXParser, FlightRecord and the classes in the utils package. The sequence of steps is summarized in the sequence diagram in Error: Reference source not found with the messages: readRecord, recordAirline, chooseRoute, modifyRecords, and writeNewTRX. This process implies reading the whole original TRX files and writing a new one which FACET takes as input instead of the original.

The message recordAirline infers the airline code from the flight number in the input file. This is so because there is no information in the original TRX file for the airline. If there is no airline with that name already registered in the AirlinesCollection in the class FACETExperiment, a new airline is created and registered. A newly registered airline tries to recover its Q-Function from the database. If there is no Q-Function recorded for this airline (i.e., this is the first execution of the experiment), the function will be created step-by-step during the simulation using the RoutesCollection and the specifications of time, origin, and destination taken from the input file.

The message chooseRoute deserves a section of its own and it will be described below.

The message modifyRecord creates a ProbabilityDistribution object. The type of the probability distribution can be “exponential” or a special type that returns a constant number (used to deactivate the push-back delay functionality). If the type is exponential, the message modifies the time stamp of the each flight according to the distribution. This
changes the structure of the original TRX file because the record will no longer be together
other records in its group (since the time stamp record groups all the flight records at the
same time), but it will be in another group. If no push-back delay is used, this message has
no effect.

The message writeNewTRX rearranges the records, if needed, and writes a new TRX
file to the disk for FACET to read it later.

3.4.2 The Configuration File

The parameters for the simulation and other modules are stored in a configuration file
described by Grammar 3.2.

Table 3.3 describes the meaning of the keys of the configuration file. The paths and
filenames must be valid in the Linux environment. The driver and server names depend on
the type of JDBC driver being used and the computer in which the databases are located.
The user and password are strings. The names of the keys are case insensitive, but the
filenames and paths are case sensitive. The values true and false are case insensitive.

3.4.3 Selection of Flightplan Routes

The simulator selects flightplan routes before the takeoff of the flights. Each pair of airports
(O/D pair) is connected by a set of alternate routes. Each O/D pair is connected by, at
least, the direct route, i.e., the GCD route. This route is the simplest and shortest route
between two airports.

The set of alternate routes is stored in the Routes table of a database (see Figure 3.14).
Each unique alternate route is identified with a unique sequence number. The routes are
represented by a flightplan string which is a sequences of navigation aids names, coordinates,
or airport codes. These routes are collected from sample TRX files with data from actual
flights. The set of alternate route also grows when the simulation finds a new route that it
did not have recorded.

The quality of the set of alternate routes has an effect in the results of the experiments.
Grammar 3.2 BNF-like grammar of the configuration file

\[
\text{grammar_file} \rightarrow \left( \text{line_comment} \mid \text{record} \right)^+ \\
\text{line_comment} \rightarrow \# \left( \text{string} \right) \\
\text{record} \rightarrow \text{verbose} = \left( \text{switch} \right) \\
\text{database} = \left( \text{driverName} \right) \left( \text{serverName} \right) \left( \text{user} \right) \left( \text{password} \right) \\
\text{stochasticity} = \text{pushback} \left( \beta \right) \text{speed} \left( \text{digit} \right)^+ \\
\text{accessibility} = \left( \text{all} \mid \text{airline} \right) \\
\text{latency} = \left( \text{digit} \right)^+ \\
\text{accuracy} = \left( \text{percent} \right) \\
\text{outputPath} = \left( \text{path} \right) \\
\text{landedLimit} = \left( \text{digit} \right)^+ \\
\text{utilizationFile} = \left( \text{filename} \right) \\
\text{arrivalsFile} = \left( \text{filename} \right) \\
\text{flightsFile} = \left( \text{filename} \right) \\
\text{logFile} = \left( \text{filename} \right) \\
\text{flyingMode} = \left( \text{DirectRoutes} \mid \text{FlightPlans} \right) \\
\text{conflictDetection} = \left( \text{file} \left( \text{filename} \right) \mid \text{socket} \left( \text{ip} \right) \left( \text{port} \right) \right) \left( \text{cparams} \right) \\
\text{inputParams} = \left( \text{mode} \right) \\
\text{adaptation} = \left( \text{true} \left( \text{learnParams} \right) \mid \text{false} \right) \\
\left( \text{mode} \right) \rightarrow \text{files} \left( \text{FACET} \mid \text{PNP} \right) \left( \text{filename} \right) \left( \text{creation} \right) \\
\text{socket} \left( \text{ip} \right) \left( \text{port} \right) \\
\text{DB} \left( \text{driverName} \right) \left( \text{serverName} \right) \left( \text{user} \right) \left( \text{password} \right) \\
\left( \text{learnParams} \right) \rightarrow \left( \varepsilon \right) \left( \lambda \right) \\
\left( \beta \right) \rightarrow \left( \text{digit} \right)^+ \left( \cdot \left( \text{digit} \right)^+ \right)^? \\
\left( \text{creation} \right) \rightarrow \left( \text{switch} \right) \\
\left( \text{cparams} \right) \rightarrow \left( \text{switch} \right) \left( \text{horizontal} \right) \left( \text{verticalAbove1900} \right) \left( \text{verticalBelow1900} \right) \\
\left( \text{switch} \right) \rightarrow \left( \text{true} \mid \text{false} \right) \\
\left( \text{horizontal} \right) \rightarrow \left( \text{digit} \right)^+ \\
\left( \text{verticalAbove1900} \right) \rightarrow \left( \text{digit} \right)^+ \\
\left( \text{verticalBelow1900} \right) \rightarrow \left( \text{digit} \right)^+ \\
\left( \varepsilon \right) \rightarrow \left( 0 \mid 1 \right) \left( \cdot \left( \text{digit} \right)^+ \right)^? \\
\left( \lambda \right) \rightarrow \left( 0 \mid 1 \right) \left( \cdot \left( \text{digit} \right)^+ \right)^? \\
\left( \text{digit} \right) \rightarrow 0-9
\]
Table 3.3: Description of the keys in the configuration file

<table>
<thead>
<tr>
<th>Key</th>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbose</td>
<td>true, false</td>
<td>If true the simulation prints more information to the standard output. With false only error messages and some information messages are printed</td>
</tr>
<tr>
<td>usingMason</td>
<td>true, false</td>
<td>Turns the on or off the multi-agent simulation platform (MASON)</td>
</tr>
<tr>
<td>database</td>
<td>driver, server, user, password</td>
<td>Defines driver, server name, user, and password to connect to the results database. The actual values of these parameters are driver and database dependent</td>
</tr>
<tr>
<td>stochasticity</td>
<td>(pushback $\beta$), (speed integer)</td>
<td>Defines expected value push back delays and the maximum absolute error for flight speeds</td>
</tr>
<tr>
<td>accessibility</td>
<td>all, airline</td>
<td>The accessibility independent variable.</td>
</tr>
<tr>
<td>latency</td>
<td>integer number</td>
<td>The latency independent variable.</td>
</tr>
<tr>
<td>accuracy</td>
<td>real number</td>
<td>The accuracy independent variable.</td>
</tr>
<tr>
<td>outputPath</td>
<td>Unix path</td>
<td>The directory (in the file system) to write the output files</td>
</tr>
<tr>
<td>utilizationFile</td>
<td>filename</td>
<td>The sector utilization file</td>
</tr>
<tr>
<td>arrivalsFile</td>
<td>filename</td>
<td>The airport arrivals file</td>
</tr>
<tr>
<td>flightsFile</td>
<td>filename</td>
<td>The flights file</td>
</tr>
<tr>
<td>landedLimit</td>
<td>integer number</td>
<td>Sets the maximum number of landed flights that FACET records during the simulation</td>
</tr>
<tr>
<td>conflictDetection</td>
<td>filename, 3 integer numbers</td>
<td>Defines output for conflict detection data, and the parameters of the conflict detection</td>
</tr>
<tr>
<td>flyingMode</td>
<td>DirectRoutes, FlightPlans</td>
<td>Determines if flights fly direct routes or flightplans</td>
</tr>
<tr>
<td>inputParams</td>
<td>type of input, params of source</td>
<td>Defines the type of input data source: file, a socket, or database. The simulation accepts FACET files (TRX format), PNP files, and database. Each with different parameters</td>
</tr>
<tr>
<td>adaptation</td>
<td>(true $\varepsilon$, learning rate), false</td>
<td>Defines if adaptation is used, and the parameters of adaptation when it is used</td>
</tr>
</tbody>
</table>

Ideally, the set should contain a sufficiently large sample of the realistic routes for all O/D pairs. However, short distance flights have less alternative routes than longer distance flights, since there are fewer navigation aids in the route or none at all. Figure 3.3 is the distribution of alternative routes. Most of the O/D pairs found in the Routes table have only the GCD route.

Airlines need to rank each one of the alternate routes to make informed selections. For that purpose, each airline keeps a Q-Function defined in Equation 2.2 on page 41 with different notation. In this function the action in Equation 2.2 is the selected route $r$. The state, $s$, in Equation 2.2 is combination of the time (the scheduled departure of the flight),
origin airport, and destination airport. Following the standard concept, the Q-Function represents the value of choosing route $r$ in state $s$. The higher the value associated to the $r \times s$ pair, the more benefits the airline obtains by selecting the route $r$.

**Algorithm 3.1 Route selection algorithm**

1: function selectRoute($s, \epsilon, Q$)

Require: $0 \leq \epsilon \leq 1$

2: $x \leftarrow \text{RAND}()$ \quad \triangleright \text{rand is uniformly distributed}

3: if $x \leq (1 - \epsilon)$ then \quad \triangleright \text{Exploit current knowledge}

4: \quad $r_s \leftarrow \arg \max_r Q(r, s)$

5: else \quad \triangleright \text{Explore new options}

6: \quad $Q_t \leftarrow Q(s) - \arg \max_r Q(r, s)$ \quad \triangleright \text{All in Q(s) except the best}

7: \quad $i \leftarrow \text{RAND}(|Q_t|)$ \quad \triangleright \text{rand is uniformly distributed}

8: \quad $r_s \leftarrow Q_t[i]$

9: end if

10: return $r_s$

11: end function

Selecting always the best option leads to suboptimal solutions because in complex systems there is no a priori information about the topology of the solution space. The current best solution could be a local optimum. The occasional selection of options that are not the current best explores the solution space better, and helps escaping from local minima. This technique is used in Local Search Algorithms [Russell and Norvig, 2003].

The situation in which the selection process always takes the best options is called exploitation [Sutton and Barto, 1998]. The situation in which the process takes less-than-best options is called exploration. The $\epsilon$-greedy algorithm (shown in Algorithm 3.1) [Sutton and Barto, 1998] combines exploration and exploitation. The parameter $\epsilon$ is the probability of randomly choosing a sub-optimal route. Low values of $\epsilon$ make this algorithm more exploitative (lines 3 and 4). Higher values of $\epsilon$ turn it more explorative (lines 6 to 8). The notation $Q(s)$ in the algorithm represents the set of alternate routes associated to the state $s$. The implementation of this algorithm must consider cases when there is only one route in $Q(s)$, when all values of $Q(r, s)$ are equal, and when there are only two routes in $Q(s)$ with different values (i.e., only one sub-optimal option).
Not all the flights are eligible to select a route. The international flights arriving in the US and the flights that are too far from the origin airport at the moment of appearing for the first time in the simulation are not eligible for the route selection process. The distanceToAirportThreshold parameter of the configuration file determines the distance threshold to consider a flight close enough to the origin airport. Only flights close enough to the origin (US) airport are subject to the flightplan route selection process.

The NAS-Wide Simulator Component: FACET

The *NAS-wide Simulator Component* simulates air traffic through the NAS. This component is implemented by NASA’s FACET. FACET has been under development and improvement for more than a decade [Bilimoria et al., 2001]. It has gained credibility as it has been used for several studies in the past [Agogino and Tumer, 2008, Calderon-Meza et al., 2009, Calderon-Meza and Sherry, 2010b, Calderon-Meza and Sherry, 2010a, Sridhar et al., 2002].

FACET has a Java GUI\(^7\) from which all the parameters and functions can be controlled. The GUI can also display the flights as they progress, their histories, their intended paths, the location of airports, airways, and navigation aids. The GUI also provides with some external applications to plan, analyze what-if conditions, produce charts, store data into a MySQL database and write queries, define filters, and create flights.

FACET has a Java API\(^8\) that allows the control some of the FACET functionality. The API consists of several Java objects that implement a Java *interface* called *CSInterface*. The actual class of the objects is not known to the user of the API, but the interfaces are documented. The interfaces of the API are:

- AircraftInterface: interface to the aircraft (flights) in the simulation
- AirportInterface: interface to the airports and their parameters
- CenterInterface: interface to the sectors in the NAS

\(^7\)GUI: Graphic User Interface
\(^8\)API: Application Program Interface
• ConflictInterface: interface to the airborne conflict detection functionality

• EnvironmentInterface: interface to weather, gravity, and terrain elevation

• FACETAPIInterface: basic FACET API interface. It is used to load FACET and start the simulations

• FlowInterface: interface to various flow modeling functions

• GUIInterface: interface to access some FACET functions and to draw on the screen

• NavigationInterface: interface to access the navigation aids database

• PlanningInterface: interface to a planning application that was added to FACET

• SectorInterface: interface to access the sectors database

• SimInterface: interface to load input data into FACET and to retrieve information about the simulation, e.g., the current time of the simulation

• SUAInterface: interface to access the Special User Airspace (SUA) database

• TrafficInterface: interface to the traffic density management function of FACET

• UtilsInterface: interface to access miscellaneous functions, e.g., measuring GCDs between two geographical points.

The core of FACET is implemented in C/C++ for the Linux environment. FACET is a physics-based simulation tool, i.e., flights are represented by a model that includes drag, thrust, lift, and weight as forces. The core of FACET does not implement any Air Traffic Control function, but it allows the user to implement it. FACET contains databases for navigation aids, coordinates, dimensions, and capacities of sectors, centers, airports, restricted areas (e.g., no-flight zones, and military bases), time zones, and performance data for many types of aircraft. It also includes algorithms for flying routes using navigation aids, or direct routes between two coordinates.
Algorithm 3.2 FACET’s ground delay program algorithm

1: procedure groundDelay
2:     for all flight ∈ flights do
3:         schArrival ← computeSchArrival(time, coordinates, flightPlan, speed) ▷ All variables in this line are members of flight
4:     end for
5:     for all airport ∈ airports do
6:         arrivals ← computeArrivals(airport) ▷ List of flights arriving at the airport
7:         airport.arrivalRate ← computeArrivalRate(airport, arrivals)
8:     if airport.arrivalRate < airport.capacity then
9:         assignGroundDelays(airport, arrivals) ▷ Assign delays to match capacity
10:     end if
11: end for
12: end procedure

FACET provides functionality to model GDPS, and the functionality is used in this dissertation. The functionality works first by predicting the arrival time of the flights (see Algorithm 3.2). The prediction is based on the flightplan (route), the filed speed, and the time and coordinates in which the flight first appears in the simulation. For each airport, FACET estimates the arrival rate based on the arrival time predictions. If the arrival rates are higher than the capacity of the airport, then the scheduled departure times are modified by introducing delays. These delays reduce the arrival rates to match it to the capacity. In this dissertation, the GDPS are used to model the manipulation of time in 4D trajectories.

FACET can output to a MySQL database and to text files. Some of the output functions can be activated via the API, but some require user intervention through the GUI. Some of these output functions are not accessible from the API (e.g., saving the file with the delays due to Ground Delay Programs can only be done from the GUI).

FACET’s functionality of airborne conflict detection is used in this dissertation. In the experiments, the Main Application component command FACET to detect airborne conflicts and to write an airborne conflicts text file. This file is processed later to compute the total number of airborne conflicts.

The conflict detection algorithm creates a safety zone around the aircraft (see Figure 3.15). The dimensions of this zone are parameters for the conflict detection algorithm. Whenever the zones of two aircraft overlap, in space and time, there is loss of separation,
Conflicts do not necessarily end in accidents, but the conflict must be timely resolved to guarantee safety. A conflict can be resolved in several ways by combining changes in the altitude, heading, or speed of the involved aircraft. In this dissertation, conflicts are detected, but not resolved since the evaluation of conflict resolution algorithms is out of the scope of the dissertation.

The ConflictsInterface of FACET does not provide a method to query the simulation about the conflicts. It only provides methods to define the detection parameters and the output file. So, the Main Application component calls the method setConflictFile of the ConflictInterface to set the filename for the airborne conflicts file. The method setConflictDetectionParameters sets the parameters of the conflict detection. The parameters include the size of the safety zone (see Figure 3.15) and the area of the NAS in which the detection is enabled.

The airborne conflicts file is a tab-separated text file that contains detailed descriptions of the airborne conflicts between aircraft. The syntax of the file is described by Grammar 3.3.

Each record of the file consists of two lines separated by a LF (line feed). Each line describes one of the two aircraft involved in the conflict. The time is given in seconds from
Grammar 3.3 BNF-like specification of the airborne conflicts file

\[
\langle \text{file} \rangle \rightarrow \langle \text{headers} \rangle \langle \text{record} \rangle^+ \\
\langle \text{headers} \rangle \rightarrow \text{Time ACID1-ACID2 Latitude Longitude Altitude Speed Heading dh/dt Length Location} \\
\langle \text{record} \rangle \rightarrow \langle \text{time} \rangle \langle \text{flight_id} \rangle \langle \text{flight_id} \rangle \langle \text{location} \rangle \langle \text{time} \rangle \rightarrow \langle \text{digit} \rangle.0 \\
\langle \text{flight_id} \rangle \rightarrow \langle \text{alpha} \rangle^+ \langle \text{digit} \rangle^+ \\
\langle \text{latitude} \rangle \rightarrow [-^? \langle \text{digit} \rangle, \langle \text{digit} \rangle^+]? \\
\langle \text{longitude} \rangle \rightarrow [-^? \langle \text{digit} \rangle, \langle \text{digit} \rangle^+]? \\
\langle \text{altitude} \rangle \rightarrow \langle \text{digit} \rangle \{1-5\} \\
\langle \text{speed} \rangle \rightarrow \langle \text{digit} \rangle \{1-3\} \\
\langle \text{heading} \rangle \rightarrow [-^? \langle \text{digit} \rangle, \langle \text{digit} \rangle^+]? \\
\langle \text{dh/dt} \rangle \rightarrow \langle 0.0 \rangle \\
\langle \text{length} \rangle \rightarrow \langle \text{digit} \rangle \{1-3\} (.\langle \text{digit} \rangle^+)\? \\
\langle \text{location} \rangle \rightarrow \text{IN} \\
\]

the start of the simulation. Latitude and longitude are given in degrees where negative values represent western or southern coordinates. The altitude is given in feet. The speed is given in knots. The heading is given in degrees where 0 is north, and 90 is west. DH/DT is the change in altitude through time of the flight. A fragment of an actual file is shown in Example 3.1.

Example 3.1 Fragment of a airborne conflicts file

```
# Time ACID1-ACID2 Latitude Longitude Altitude Speed Heading dh/dt Length Location
38760.0 XE2004 YV7149 41.80084246707898 -86.51893320468767 30000 435 213.91387711873784 0.0 .78.54779071447108 IN
41.758558212275155 -86.6157590671127 29000 414 283.5053351404967 0.0 13.801516519887304 IN
39660.0 CO1121 DL921 42.31803847319608 -71.09702582059433 33500 439 235.84652648773772 0.0 4.880642488351678 IN
42.33726680201904 -71.15752817562709 33700 415 256.77519661499946 0.0 6.923344569359298 IN
```

Conceptually, the airborne conflicts file defines a set \( C_{alt} \) of pairs \( \langle t_i, (f_a, f_b) \rangle \), each one representing a detected conflict between flights \( f_a \) and \( f_b \) at time \( t_i \). The other data in the file is not relevant for this dissertation. The sub-index for the time is delimited by \( 1 \leq i \leq T \) where \( T \) is the greatest time stamp of the simulation, i.e., \( T = 43200 \) since the simulation...
only simulates 24 hours of operation. Another observation is that \((f_a, f_b)\) is not an ordered pair, so \((f_a, f_b) \neq (f_b, f_a)\).

FACET checks for the occurrence of conflicts at each time step of the simulation without recording history. So if one conflict lasts for more than one time step the conflict is reported as many times as time steps it lasts. Therefore, \(C_{all}\) contains several pairs \(\langle t_{i1}, f_a, f_b \rangle\), \(\langle t_{i2}, f_a, f_b \rangle\), \(\langle t_{i3}, f_a, f_b \rangle\) where \(t_{i1} = t_{i2} - step\) and \(t_{i2} = t_{i3} - step\). To avoid multiple counting the Main Application component summarizes the airborne conflicts file at the end of an execution of the simulation to compute the total number of airborne conflicts and the duration of each conflict. It also increments the conflict counts of the corresponding flights and airlines. The values for conflicts are used as metrics during the adaptation and the analysis of results.

The summarization process can be defined as follows:

Be \(NC = |C|\) the total number of airborne conflicts detected, where \(C\) is a set constructed as follows.

Let the set of detected conflicts at time \(t_i\) be

\[
C_0 = \emptyset
\]
\[
C_i = \{(f_a, f_b)|\langle t_i, (f_a, f_b) \rangle \in C_{all}\}
\]

Then

\[
C = \bigcup_{i=1}^{T}\{(t_i, (f_a, f_b))|(f_a, f_b) \in (C_i - C_{i-1})\}
\]

The class ConflictsProcessor summarizes the airborne conflicts file. For that purpose the class uses the ConflictsParser class and follows the algorithm defined in Algorithm 3.3.
Algorithm 3.3 Parsing process of the airborne conflicts file

1: function parseConflictsFile()
2:   conflicts ← ∅
3:   parser.skipHeader()
4:   if parser.moreRecords() then
5:     r ← parser.readRecord() ▷ Records contain time of conflict, flights 1 and 2
6:     if ⟨t, r.t − 1, r.f1, r.f2⟩ ∈ conflicts then
7:       conflicts ← (conflicts − {⟨t, r.t − 1, r.f1, r.f2⟩}) ∪ {⟨t, r.t, r.f1, r.f2⟩} ▷
8:         Update the end time of the conflict
9:     else ▷ No active conflict between f1 and f2 recorded
10:        conflicts ← conflicts ∪ {⟨r.t, r.t, r.f1, r.f2⟩}
11:   end if
12: end if
13: return ‖conflicts‖
14: end function

Computation of the Performance Metrics

There are several types of performance metrics. Flights, airlines, and the system have corresponding vectors of performance metrics.

The metrics for the flights are computed (except for the number of conflicts) in the step method of the Flight class (and agent in MASON). The AircraftInterface provides information about a flight:

- The fuel-burn \( f_i \)
- The actual departure time \( t^a_{d_i} \)
- The actual landing time \( t^a_{l_i} \). Only if the flight is already landed \( (ad_{ij}) \)
- The current coordinates, \( (lat_a, long_a) \)
- The current speed, \( s_a \).

The actual departure time, \( t^a_{d_i} \), is subtracted from the scheduled departure time, \( t^s_{d_i} \), to obtain the \textit{departure delay}, \( gd_{ij} \), of the flight. In this case, the scheduled departure time is approximated by the first time in which the flight appears in the input file of the simulation if the flight was either not flying at that time or if it was close enough to the origin airport.
Algorithm 3.4 Estimation of flight arrival time

1: function estimateScheduledArrival(position, speed, fp)

Require: position is the current coordinates of the flight

Require: fp is a sequence of points (coordinates) \( p_1, p_2, \ldots, p_N \)

2: \( p_i \leftarrow \text{findClosestFlightPlanPoint}(\text{position}, \text{fp}) \) \( \triangleright \) First estimate the distance to fly

3: \( d_i \leftarrow \text{GDC}(\text{position}, p_i) \)

4: \( d_{i+1} \leftarrow \text{GDC}(\text{position}, p_{i+1}) \)

5: if \( d_i \leq d_{i+1} \) then \( \triangleright \) It is shorter to fly first to \( p_{i+1} \) than to \( p_i \)

6: \( d \leftarrow d_{i+1} \)

7: \( j \leftarrow i + 2 \)

8: else

9: \( d \leftarrow d_i \)

10: \( j \leftarrow i + 1 \)

11: end if

\( \triangleright \) Cumulate the distance from the next point of the flightplan \( p_j \) to the end of the flightplan

12: \( d \leftarrow d + \sum_{i=j}^{N-1} \text{GDC}(p_i, p_{i+1}) \) \( \triangleright \) Assume speed is constant throughout the flight

13: return \( \leftarrow \text{schDepTime} + d / \text{speed} \)

14: end function

The actual landing time, \( t^a_t \), is substracted from the scheduled arrival time, \( t^s_t \), to obtain the arrival delay, \( ad_{ij} \), of the flight. The scheduled arrival time is not included in the input file of the simulation, i.e., this time must be estimated. The Algorithm 3.4 shows how to the scheduled arrival time is estimated from the input files.

At each time step, the GCD between the current coordinates of the flight, \((\text{lat}^a, \text{long}^a)\), and the previous coordinates, \((\text{lat}^p, \text{long}^p)\), is computed and accumulated in the distance flown, \( d_{ij} \), of the flight.

The class SectorUtilizationProcessor defines and maintains, with the help of the Sector-Interface, a function \( n : t_i \times s_j \rightarrow N \), to represent the number of aircraft in sector \( s_j \) at time \( t_i \). The capacity of the sector, \( c_j \), is provided by the SectorInterface. One more flight performance metric is related to the en-route congestion the flight experiences. The metric measures the percentage of sectors crossed by the flight that were congested, \( %s_{ij} \). This metric is only available when airlines have access to SWIM. Figure 3.16 shows an example of computation of this metric for a simple case.
At each time step, the Flight objects determine if they cross to another sector from the previous time step, with this information flights update the count of sectors they crossed. With the SectorInterface, flights can also determine how many flights are in their sectors (i.e., the function $n$ described above) and the capacity of the current sector, $c_j$. So flights also count the number of congested sectors they cross. When the flight lands, it computes the $\% s_{ij}$ with all data it collected during its en-route period.

Flights are also assigned the number of conflicts they are involved in, $c_{ij}$. This is done in the end of the execution when the airborne conflicts file is processed (previous section).

The airline performance metrics are computed in the end of the execution.

- The airline fuel-burn $f_a = \sum_i f_i^a$ for all the flights of the airline

- The airline departure delay $gd_j = \sum_i gd_{ij}$

- The airline arrival delay $ad_j = \sum_i ad_{ij}$

- The airline conflicts, $c_j$, for all the flights of the airline involved in conflicts
• The airline distance, \( d_j = \sum_i d_{ij} \)

The system (NAS) performance metrics are computed at the end of the execution. The process of computation starts with a list \( L_{all} \) of pairs \( (t_i, a) \), each one representing an arrival at airport \( a \) at time \( t_i \). This list is maintained by the CongestionProcessor class.

The first NAS performance metric is the percentage of the time periods in which at least one airport at or above its arrival rate, \( \%OSch \). The arrival rates of the airports are parameters of the NAS (i.e., they are modeled in this dissertation as parameters of the experiment). The \( \%OSch \) measures the congestion of the arrival airports, and it is computed as follows:

Be \( t_I \) and \( t_F \) the initial and final times of the simulation, and be \( \Delta t \) the time step. There are \( K \) steps of duration \( \Delta t \).

\[
\frac{t_F - t_I}{\Delta t} = K
\]

Let there be the binary function

\[
p(t, a) = \begin{cases} 
1 & \text{if } (t, a) \in L_{all} \\
0 & \text{if } (t, a) \notin L_{all} 
\end{cases}
\]

The number of arrivals in the period \( k \) at airport \( a \) is defined by

\[
n_a(k, a) = \sum_{t_i} p(t_i, a) \text{ for } 0 \leq k \leq K \text{ and } t_I + k \times \Delta t \leq t_i \leq t_I + (k + 1) \times \Delta t
\]

if the arrival rate of an airport for a period \( \Delta t \) is given by \( c(a) \) then the arrival rate ratio for an airport at period \( k \) is

88
the metric for the whole system is the percentage of the total number of periods during which at least one airport is at or above its “declared” arrival rate as shown in Equation 3.1 where $H$ is the Heaviside step function.

$$o_a = \frac{1}{K} \sum_{k,a} H(ARR_{ratio}(k,a) - 1)$$

The second NAS performance metric is the percent of simulated time, $T$, in which there is at least one sector at or above its MAP$^9$, %OL. It measures sector congestion, and it is computed as follows:

Be

$$o(t_i, s_j) = \begin{cases} 
1 & \text{if } n(t_i, s_j) \geq c(s_j) \\
0 & \text{if } n(t_i, s_j) < c(s_j) 
\end{cases}$$

then

$$per_o = \frac{\sum_i o(t_i, s_j)}{T}$$

where $H(x)$ is the Heaviside step function.

Four more NAS performance metrics are defined as follows:

- The total fuel-burn for each airline $f = \sum_i f_j$ for all the flights of the airline

$^9$MAP: Monitor Alert Parameters, a metric of the capacity of a sector in a period of time
Table 3.4: Performance metrics used in the simulations

<table>
<thead>
<tr>
<th>Metric</th>
<th>Symbol</th>
<th>Performance vector</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fuel burn</td>
<td>$f_{ij}$</td>
<td>Flight</td>
<td>Flight fuel of flight $I$ from airline $j$</td>
</tr>
<tr>
<td>Departure delay</td>
<td>$gd_{ij}$</td>
<td>Flight</td>
<td>Departure delay of flight $I$ from airline $j$</td>
</tr>
<tr>
<td>Arrival delay</td>
<td>$ad_{ij}$</td>
<td>Flight</td>
<td>Arrival delay of flight $I$ from airline $j$</td>
</tr>
<tr>
<td>Distance</td>
<td>$d_{ij}$</td>
<td>Flight</td>
<td>Distance of flight $I$ from airline $j$</td>
</tr>
<tr>
<td>Conflicts</td>
<td>$c_{ij}$</td>
<td>Flight</td>
<td>Airborne conflicts in which flight $I$ of airline $j$ is involved. This metrics is only available if SWIM is present.</td>
</tr>
<tr>
<td>% of congested sectors crossed</td>
<td>$%s_{ij}$</td>
<td>Flight</td>
<td>Percentage of congested sector flight $I$ from airline $j$ crossed in its flight. This metrics is only available if SWIM is present.</td>
</tr>
<tr>
<td>Airline fuel burn</td>
<td>$f_j$</td>
<td>Airline</td>
<td>Fuel burn of airline $j$</td>
</tr>
<tr>
<td>Airline departure delay</td>
<td>$gd_j$</td>
<td>Airline</td>
<td>Departure delay of airline $j$</td>
</tr>
<tr>
<td>Airline arrival delay</td>
<td>$ad_j$</td>
<td>Airline</td>
<td>Arrival delay of airline $j$</td>
</tr>
<tr>
<td>Airline airborne conflicts</td>
<td>$c_j$</td>
<td>Airline</td>
<td>Airborne conflicts in which flights from airline $j$ are involved</td>
</tr>
<tr>
<td>Airline distance</td>
<td>$d_j$</td>
<td>Airline</td>
<td>Distance of airline $j$</td>
</tr>
<tr>
<td>Total fuel burn</td>
<td>$f$</td>
<td>NAS</td>
<td>The total fuel burn of the system (NAS)</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>$gd$</td>
<td>NAS</td>
<td>The total departure delay of the system</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>$ad$</td>
<td>NAS</td>
<td>The total arrival delay of the system</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>$c_T$</td>
<td>NAS</td>
<td>The total number of conflicts of the system</td>
</tr>
<tr>
<td>% overloaded sectors</td>
<td>$%OL$</td>
<td>NAS</td>
<td>The percentage of time steps in which at least one sector is above its MAP value</td>
</tr>
<tr>
<td>% congested airport</td>
<td>$%OSch$</td>
<td>NAS</td>
<td>The percentage of time steps in which at least one airport is receiving more arrivals than its VFR capacity value</td>
</tr>
</tbody>
</table>

- The total departure delay for each airline $gd = \sum_j gd_j$

- The total arrival delay for each airline $ad = \sum_j ad_j$

- The total conflicts, $c_T = \sum_j c_j$

Table 3.4 summarizes the metrics explained above and determines their corresponding performance vector. In this dissertation, performance is always a multi-variable function [Deb, 1999, Fonseca and Fleming, 1995].
3.4.4 Adaptation of Airline Decision-Making

The Adapt behaviors function evaluates the effects, on the performance, of the route choices made by the airlines during the simulation and adapts the route selection behavior to achieve better airline performance.

This function adapts the pre-take-off route selection behavior of the airlines. The behavior of each airline is represented by a multi-variable function called Q-Function. The Q-Functions relate route choices to the value of those choices for the airline. The value of the choice reflects the performance obtained by the airline and the system due to that choice.

Flight schedules in the NAS are repetitive. Most of the airlines only make small changes in the time of departure or arrival of the flights. Changes in the number of flights per day or in the origin / destination pairs are less frequent. This slow-changing process allows the agents to try different routes, measure performance, and use the performance information of previous instances of the flight to choose the best route. The actual selection of the best route is still a hard problem since there are several stochastic factors involved. The decision must be supported by models that consider the stochastic nature of the process [Burdun and Parfentyev, 1999].

In this dissertation, the behavior of the AOCs is abstracted as a revision of past decisions in similar situations. The results of previous executions of the simulation, with the same global schedule, are stored in a data repository. When a flight is about to enter the simulation, the previous results will be retrieved and the “best” route, i.e., the route that has the most benefits for the airline, will be used again. The measurement of performance requires the consideration of multiple objectives. Section 2.6.4 on page 42 explained the techniques to compare multi-objective performance. This dissertation uses a modified definition of domination to guide the learning process of the agents and implement adaptation.

With this type of adaptation, the agents are rational [Russell and Norvig, 2003], they seek their own benefit. There is no cooperation between them to achieve global benefits. This is the way the current NAS is regulated: no cooperation is allowed between airlines.
Further analysis will be required to determine global patterns and possible equilibriums in the system that result from the rationality of these agents.

**Algorithm 3.5** Update the Q-Functions for the airlines

```latex
1: \textbf{procedure} \text{updateQ}(qRecord, R) \triangleright qRecord is one element of $Q : r \times s \to \mathbb{R}$
2: \hspace{1em} $s \leftarrow qRecord.s$
3: \hspace{1em} $r \leftarrow qRecord.r$
4: \hspace{1em} $qRecord.Q(r, s) \leftarrow (1 - \lambda)qRecord.Q(r, s) + \lambda R \triangleright$ Update the “value” for route $r$ at state $s$
5: \textbf{end procedure}
```

The Algorithm 3.5 describes the modifications of the Q-Function of each airline for each route it chose during the execution. The line 4 of the Algorithm 3.5 is a simplification of the Equation 2.2 on page 41. The simplification is valid because the selection of a route (action) does not lead the system to a different state. In other words, the system modeled here does not exhibit a sequence of states, but only a single decision per state. Therefore, the part of Equation 2.2 that deals with the future rewards is not needed in this problem. A high value of $\lambda$ gives more importance to the rewards obtained by current decisions than to the rewards obtained in the past in similar situations. The effect is the opposite for low values of $\lambda$.

The Algorithm 3.6 describes the adaptation process used in this research by each AOC autonomously. The combination of Algorithms 3.5, 3.6, and 3.1 (on page 76) is called $\varepsilon$-greedy Q learner.

**Algorithm 3.6** Algorithm to adapt the airline behavior

```latex
1: \textbf{procedure} \text{ADAPT}
2: \hspace{1em} \textbf{for all} flight of an airline \textbf{do}
3: \hspace{2em} value $\leftarrow$ \text{COMPUTE}\text{REWARD}(flight)
4: \hspace{2em} choice $\leftarrow$ \text{FIND}\text{Q}\text{RECORD}(flight) \triangleright$ Q-Record corresponding to flight
5: \hspace{2em} \text{UPDATEQ}(choice, value)
6: \textbf{end for}
7: \textbf{end procedure}
```
Rewards Computation: the Implementation of Modified Domination

A reward is a measure of the value of a decision (i.e., flightplan route selection) made by an airline. The value of the decision depends on the performance achieved by the airline due to that decision. The performance of a flight, "i", is a vector: \( \eta_i = \langle m_1, m_2, m_3, \ldots, m_n \rangle \).

The elements of the vector are values of performance metrics from the first 6 rows of Table 3.4 or the first 4 if no global information is available to the airlines.

Section 2.6.4 explained the limitations of the domination in learning processes. The approach taken in this dissertation is to modify the definition of domination to better reflect the progress in the learning process. The result is a kind of domination operator that is discrete, but not binary. This definition of domination reflects how much better a selection performs compared to another selection (see Equation 3.3).

Let \( A = \langle a_1, a_2, \ldots, a_n \rangle \) and
\[
B = \langle b_1, b_2, \ldots, b_n \rangle
\]
(3.3)

\[
dom(A, B) = |\{a_k | \forall k \text{ such that } a_k \geq b_k \}|
\]

When vector A is “better” then vector B in m out of the n elements \( \dom(A, B) = m \).

There are \( (n+1) \) possible integer values for \( \dom \) from 0 to n in inclusive. A value of 0 indicates that A is not better than B in any of the metrics. A value of n indicates that the A dominates B (i.e., it performs better or the same in all metrics).

This discretization of domination helps the learning process move toward progressively better choices.

The computation of the reward for a route selection of an airline is described in Algorithm 3.7. With Algorithm 3.7, if a route selection performs better in all the metrics than all the past selections the reward is n. If the route selection performs worse in all the metrics than all the other past selections the reward is 0. The reward is a ranking of the route selection compared to the past selections for similar situations (i.e., around the same
Algorithm 3.7 Computation of rewards

1: function computeRewards(flight)
2: time ← flight.sch_dep
3: origin ← flight.origin
4: dest ← flight.dest
5: $B \leftarrow \text{pastMetrics}(\text{airline}, \text{time}, \text{origin}, \text{dest})$  \Comment{List of the $K$ past metrics}
6: $A \leftarrow \text{current metrics}$
7: $d \leftarrow \sum_{i} \text{dom}(A, B_i)$
8: return $d/K$
9: end function

departure time, same origin and destination).

The function readPastMetrics has two different behaviors. When airlines only have access to their own information, the metrics for conflicts, $c_{ij}$, and for the percentage of congested sectors crossed, $%s_{ij}$, are not available to the airline (see Table 3.4 on page 3.4). When SWIM is present, airlines have access to those two metrics and to records for similar situations from other airlines, too.

3.5 Design of Experiments

For these experiments, a day of operations in the NAS starts at 8:00 am UTC and ends the next day at 7:59 am UTC. The typical demand for one day of the NAS is about 60,000 flights including domestic, international, general aviation, and other non-commercial operations. The simulator can repeatedly simulate one day of operations in the NAS, i.e., this is a simulated day. The sequence diagram of Figure 3.12 shows the messages occurring in the simulator for a single simulated day.

The goal of the experiments is to test the six null hypotheses (see section 1.4 on page 7) enumerated as follows:

- $H_0^1$: The availability of system-wide information used by adaptable AOCs for pre-departure flightplan route selection results in decreased or equal NAS performance
• $H^2_0$: The availability of system-wide information used by adaptable AOCs for pre-departure flightplan route selection results in increased or equal variation in the performance metrics

• $H^3_0$: Latency in the communication of information used by adaptable AOCs for pre-departure flightplan route selection results in increased or equal NAS performance

• $H^4_0$: Latency in the communication of information used by adaptable AOCs for pre-departure flightplan route selection results in decreased or equal variation in the performance metrics

• $H^5_0$: Inaccuracies in the information used by adaptable AOCs for pre-departure flightplan route selection result in increased or equal NAS performance

• $H^6_0$: Inaccuracies in the information used by adaptable AOCs for pre-departure flightplan route selection result decreased or equal variation in the performance metrics.

These hypotheses define three independent variables or factors for the experiment: data accessibility, data latency, data inaccuracy$^{10}$. The three variables could take a range of values in the real system, but they are limited to two values for the experiment. The following list explains the variables and their possible values.

• Data availability. This variable can take two values: Global, and Local. When the value is Global the adaptation process of the airlines has access to metrics from all the airlines and metrics supported by SWIM. When the value is Local, the adaptation process has only access to metrics not supported by SWIM and to historic records of the particular airline.

• Data latency. The term latency is used in this dissertation as a synonym of delay. This variable can take two values: Real-time, and Delayed. When the value is Real-time the adaptation process of the airlines receives data from the current and all the

$^{10}$Error in the data caused by the presence of noise.
previous executions. When the value is “Delayed” the adaptation process receives data from the current execution and from executions that are at least N steps older than the current one. The value of N in this case is fixed to 2. So the adaptation is using historic data that are delayed one period.

- **Data accuracy.** This variable can take two values: *Accurate*, and *Noisy*. The term *noise*\(^\text{11}\) is used in this dissertation as a synonym of corruption in the data. When the value is Accurate the adaptation process receives the historic values from database without modification. When the value is Noisy the adaptation process receives values with errors. The magnitude of the error is normally distributed with zero mean and standard deviation that equals a given percentage of the value. The percentage is a parameter of the simulation that is defined in the configuration file of the simulation.

The *response* or *dependent variable* of the experiment is a vector of performance metrics. There is one performance vector for the NAS, one for each individual airline, and one for each flight or route selection (see Table 3.4 on page 88). Only the performance vectors for the flights are used for the adaptation process of the airlines. Airlines and NAS performance vectors are used to compare the results of airlines with different strategies and to evaluate the results of the simulation as a whole.

The flightplan route selection problem is similar to an *n-armed bandit problem*\(^\text{12}\), in which an exploration phase is recommended to learn the behavior of the “bandit”. This exploration period is followed by an exploitation period. This dissertation uses the \(\varepsilon - \text{decreasing}\) strategy to approach this problem[Sutton and Barto, 1998], because it reduces the execution time during the exploration period, is simple to implement and automate, and allows an earlier observation of the benefits provided by the learning process.

An experiment typically starts without any knowledge. The no knowledge state is represented by an empty Q-Function or a Q-Function with all its values set to zero. Knowledge is gained through time by repeating the simulations. During this knowledge acquisition

\(^{11}\)In this context, it is the result of super-imposing a normal distribution with zero mean and \(\sigma\) standard deviation on the actual measured value. It is expressed as \(3\sigma = \%\) of actual measured value

\(^{12}\)See Russell and Norvig[Russell and Norvig, 2003], section 21.3 for an explanation
process, the performance of the NAS and the airlines improves with each simulated day. After some simulated days the learning process reaches a steady-state - also called stable state - in which changes in the ranks of the routes are less frequent. In the steady-state of the learning process, the performance of the system reaches a value that is close to fix, but there is some variation. Even when the learning process is in its steady-state the system remains learning since $\varepsilon$ is always kept greater than 0. Two criteria are defined to determine when the learning process reaches the steady-state:

- The percentage of Q-Function records that have a non-zero value must reach a given threshold. This percentage is a metric for the amount of knowledge acquired by an AOC. A threshold of 60% is established for the experiments.

- The percentage of Q-Records in one day that significantly changed from their previous values. A change in the value of a Q-Record is considered significant when it exceeds 10% of its previous value. In one day, about 52,000 flights, from the total 67,000, take part in the route selection process. A threshold of 10% is set for the experiments.

A simulation is in steady-state when simultaneously meets these two criteria for the first time. For all the experiments, the values of the dependent variables are considered for analysis only after the simulation is in steady-state. Each simulated day after that threshold is a sample from the universe of possible outcomes of the simulation for the same schedule (same day of operations).

Table 3.5 is the design of experiments to test the hypotheses. The design only includes the combinations of factors that help to test the hypotheses. The comparison of experiments #1 and #2 allows for testing hypotheses #1 and #2. The comparison of experiments #1 and #3, and #2 and #4 allows for testing hypotheses #3 and #4. The comparison of experiment #1 and #5, and #2 and #6 allows for testing hypotheses #5 and #6.

Though the simulator can consider, weather (e.g., winds), push-back delays, and speed errors for the flights, these disturbances are not included in these experiments, i.e., the NAS is deterministic, and can be included in future experiments.
Table 3.5: Design of experiments

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Independent variables</th>
<th>Data availability</th>
<th>Data latency</th>
<th>Data accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Global</td>
<td>Real-time</td>
<td>Accurate</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Local</td>
<td>Real-time</td>
<td>Accurate</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Global</td>
<td>Delayed</td>
<td>Accurate</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Local</td>
<td>Delayed</td>
<td>Accurate</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Global</td>
<td>Real-time</td>
<td>Noisy</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Local</td>
<td>Real-time</td>
<td>Noisy</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.6: VFR Airport Arrival Rates (AAR) used in the simulations for the OEP-35 airports

<table>
<thead>
<tr>
<th>Airport ICAO code</th>
<th>Airport Arrival Rate (Moves per hour)</th>
<th>Airport ICAO code</th>
<th>Airport Arrival Rate (Moves per hour)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KATL</td>
<td>80</td>
<td>KLGA</td>
<td>40</td>
</tr>
<tr>
<td>KBOS</td>
<td>60</td>
<td>KMCO</td>
<td>52</td>
</tr>
<tr>
<td>KBWI</td>
<td>40</td>
<td>KMDW</td>
<td>32</td>
</tr>
<tr>
<td>KCLE</td>
<td>40</td>
<td>KMEM</td>
<td>80</td>
</tr>
<tr>
<td>KCLT</td>
<td>60</td>
<td>KMIA</td>
<td>68</td>
</tr>
<tr>
<td>KCVG</td>
<td>72</td>
<td>K MSP</td>
<td>52</td>
</tr>
<tr>
<td>KDCA</td>
<td>44</td>
<td>KORD</td>
<td>80</td>
</tr>
<tr>
<td>KDEN</td>
<td>120</td>
<td>KPDX</td>
<td>36</td>
</tr>
<tr>
<td>KDFW</td>
<td>120</td>
<td>KPHL</td>
<td>52</td>
</tr>
<tr>
<td>KDTW</td>
<td>60</td>
<td>KPHX</td>
<td>72</td>
</tr>
<tr>
<td>KEWR</td>
<td>40</td>
<td>KPIT</td>
<td>80</td>
</tr>
<tr>
<td>KFLL</td>
<td>44</td>
<td>KSAN</td>
<td>28</td>
</tr>
<tr>
<td>KHNL</td>
<td>40</td>
<td>KSEA</td>
<td>36</td>
</tr>
<tr>
<td>KIAD</td>
<td>54</td>
<td>KIAH</td>
<td>72</td>
</tr>
<tr>
<td>KSFO</td>
<td>60</td>
<td>KJFK</td>
<td>44</td>
</tr>
<tr>
<td>KSAC</td>
<td>44</td>
<td>KLAS</td>
<td>52</td>
</tr>
<tr>
<td>KSTL</td>
<td>52</td>
<td>KLAX</td>
<td>84</td>
</tr>
<tr>
<td>KTPA</td>
<td>28</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The airport arrival rates are the number of arrivals at an airport per period of time. There are several periods of time used to measure the rates; 15 minutes, and 1 hour are two usual periods of time. FACET’s default airport arrival rates are infinite: any number of arrivals can take place simultaneously. In this dissertation, the arrival rates of the 35 busiest airports\(^{13}\) in the United States are set to their VFR\(^{14}\) values (see Table 3.6).

All sectors are set to the VFR Monitor Alert Parameter (MAP) values FACET has in

---

\(^{13}\) Also called OEP-35 airports.

\(^{14}\) VFR: Visual Flying Rules. Used when weather is not causing negative effects.
its internal database. Due to the number of sectors in the NAS (about 1,500), the detailed list of sectors and their MAP values is not included in this dissertation.

After the route selection, flights actually take off and fly their routes at the altitude and speed specified in the inputs. The takeoff time of the flights could be delayed if the predicted arrival rate of the destination airport is greater than the VFR value set for the airport.

### 3.6 Analysis of Results

When an experiment starts, the AOCs select routes randomly. As the simulation progresses, the AOCs acquire knowledge and start using a greedier strategy to select routes. After some simulated days, the learning process of the AOCs reaches a steady-state (see section 3.5), and continues for some more executions. The first step of the analysis is a comparison of the system performance metrics obtained at the first simulated day (i.e., random route selection) and at the day in which steady-state is reached. The second step is to group the system performance metrics obtained during the steady-state to form samples. The analysis of the samples focuses on testing the hypotheses presented the Design of Experiments (section 3.5), which is done on the samples.

The analysis of the samples is illustrated in Figure 3.17 and explained as follows:

1. Many hypothesis testing algorithms are available for normally distributed data samples. The power of the tests depends on how close the data samples are to normality. Other hypothesis testing algorithms are available for non-normally distributed data samples. The distinction between normally distributed and non-normally distributed data samples justifies the requirement for normality tests, also known as goodness-of-fit tests. Statisticians have proposed several algorithms to determine normality of data samples. Some examples of normality tests for the composite hypothesis\(^{15}\) are D’Agostino’s K-squared, Anderson-Darling’s, Cramer-von Mises’s, Lilliefors’s

\(^{15}\)The hypothesis for normality is composite when the mean and variance of the population are unknown and must be estimated from the sample.
Figure 3.17: Flow chart of the analysis of results.
(Kolmogorov-Smirnov), Pearson’s ($\chi^2$), Shapiro-Francia’s, and Shapiro-Wilk’s. Each test has its own properties, weaknesses, and power. The decision of which test to use is not trivial and the recommendations are contradictory in many cases. This analysis does not consider tests for simple hypothesis for normality since the mean and variance of the population are not known in the experiments. After the experiment has reached steady-state, the first step of the analysis is to test for normality of the distribution for each metric of each experiment using Shapiro-Wilk’s test. The Shapiro-Wilk’s test is used because it works well for mid-size samples and it is power is acceptably high compared to the other tests.

2. Compare the samples of two experiments as indicated in the following steps.

3. Determine if the variances of the each metric are statistically equal between experiments. This is done using a two-sided F-test when the samples are both normally distributed, and with the Levene’s test if at least one of the samples is not normally distributed. The reason for using different tests is that the F-test makes the assumption of normality in the samples to guarantee its power and correctness. However, Levene’s test does not assume any distribution of the samples. Hence Levene’s is more powerful than the F-test when the samples are not normal, but when normality is known the F-test is better. The information of equality of variances is useful to decide how to compare the means of the metrics distributions.

4. For the samples that are normally distributed, the comparison of the means is done with the one-side unpaired t-test. The t-test is based on the assumption of normality of the samples. The test considers the information from the comparison of the variances as a parameter. If the variances are equal, then the standard two-sample unpaired t-test is used. The Welch’s two-sample unpaired t-test is used if the variances are different, since this test takes into account (i.e., compensates for) the difference. The one-side test is used because the hypotheses search for one-sided differences in the means. The tests are unpaired because there is no relation between days of different
experiments and because the samples are not of the same sizes as defined by the simulated day at which the simulations reach steady-state.

5. For the samples that are not normally distributed, the comparison of the means is done with the non-parametric one-sided unpaired Wilcoxon-test. Being non-parametric, Wilcoxon test is adequate to compare two samples which distributions are not normal.

6. The results from the third, fourth, and fifth steps are used to accept or reject the null hypotheses for the difference of the mean of each metric.

7. The hypotheses about the variances are tested with the one-sided F-test for all the metrics, because the Levene’s test is not defined for one-sided tests. The tests are one-sided because the hypotheses to be tested are one-sided.

As indicated in 3.17, the steps 2 to 7 are repeated for each of the comparisons of all the samples, i.e.: between experiments #1 and #2, #3 and #1, #4 and #2, #5 and #1, #6 and #2.
Chapter 4: Evaluation of the Impact of SWIM on the Performance of the NAS in the Presence of Adaptive Airline Flightplan Route Selection

This chapter presents the results of the experiments to evaluate the impact of SWIM on the performance of the NAS in the presence of adaptive airline flightplan route selection. The results were obtained from the integration of an existing NAS-wide simulator, a multi-agent simulator, a database, and the implementation of airline, i.e., AOC, adaptable behavior as described in the previous chapter.

This chapter emphasis is on tests of the stated hypotheses, and discussion of the results of the tests. The testing requires the comparison of results of pairs of experiments to determine the effect of independent variables. The results are characterized by their statistical significance. Economic considerations of the results are left for future work (see chapter 5.2 on page 160).

The chapter is organized as follows: the first section determines when the experiments reach steady-state and determine the size of the samples to be used in further tasks (see section 3.5). The second section determines the normality of the samples in order to decide which statistical test to use for testing the hypotheses. The third section describes the actual comparison between experiments and tests the hypotheses accordingly.

4.1 Stability of the Learning Process

In the beginning of an experiment the AOCs have no knowledge of the system. This is represented by having no Q-Records or having all the Q-Records initialized to zero.

The experiment has an initial exploration period in which the AOCs gain knowledge. In this period, \( \varepsilon \) determines the probability of selecting routes randomly instead of selecting
the route best-so-far. It is during this exploration period that $\varepsilon$ decreases from its initial value of 1.0 to its final value of 0.2 in fixed decrements of 0.05 each simulated day.

The two stability criteria defined in section 3.5, on page 92, determine the start of the learning process’s steady-state, and hence the start of the experiment’s exploitation period. Only the outcomes obtained during the exploitation period are considered in the analysis of results of an experiment.

Figure 4.1 shows the change of $\varepsilon$, the percentage of non-zero Q-Records, and the percentage of Q-Records that changed 10% or more from their previous values. The 60% of non-zero Q-Records is reached at the 14th simulated day. The parameter $\varepsilon$ reaches 0.2 on the 17th simulated day. Approximately 51,000 flights are involved in the route selection process each simulated day, consequently only 51,000 Q-Records could change every day. The 17th simulated day is the first time in which at most 5,100 Q-Records changed in one day. In summary, the 17th simulated day marks the start of the exploitation period for experiment #1, and the start of the simulated days in which the learning process can be considered stable.

Figures similar to Figure 4.1 can be constructed for all the other experiments, but the results are best summarized in Table 4.1 which contains the number of simulated days in which the experiments reach the steady-state. The table suggests that the changes in the independent variables affect the learning process in two ways. First, the speed of knowledge acquisition changes between experiments as reflected by the different number of days the experiments need to have 60% of the Q-Records with a non-zero value. The knowledge acquisition is faster when the information is real-time, than when it is delayed. The accuracy of the information and its scope do not show an effect in this case, which might be explained by the behavior of the flightplan route selection process. Since the selection is based on historic records and the information is delayed one day, the airlines are unable to select their best routes until the third day of operations. Before airlines start receiving historic data they select routes randomly. A random selection does not guarantee high rewards for the routes, therefore the best routes are discovered later in the simulation.
The effect of one day of delay in the availability of information is reflected in three days of delay in the knowledge acquisition.

The second effect of the independent variables in the knowledge acquisition process is that the scope of the information, its accuracy, and latency affect the time it takes to meet the second stability criterion. When the information is local (2, 4, 6) it takes from 1 to 7 more simulated days to meet the criterion, than when the information is global (1, 3, 5). Latency increases the number of days to meet the criterion from 4 to 8 simulated days when compared to cases with real time information.

Accuracy increases the number of days to meet the criterion from 2 to 11 simulated when compared to cases with accurate information. The combination of independent variables that results in the shortest times to meet the stability criteria is global, real-time, and accurate information. The combination that results in the longest times is local, delayed,
Table 4.1: Number of simulated days needed to meet the stability criteria and data sample sizes

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Reaches 60% of non-zero q-records (Simulated day)</th>
<th>Reaches 10% of q-records changing 10% or less (Simulated day)</th>
<th>Sample size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
<td>17</td>
<td>64</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>18</td>
<td>63</td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td>21</td>
<td>60</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>26</td>
<td>58</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>23</td>
<td>58</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>32</td>
<td>48</td>
</tr>
</tbody>
</table>

and noisy information.

From the simulated day in which an experiment meets the two stability criteria the outputs of the simulation become elements of the sample data. Table 4.1 includes a third column with the data sample size for each experiment.

4.2 Normality of the Data Samples

The normality tests, and all further analysis steps, consider only results obtained after the learning process reaches stability as indicated in Table 4.1 and explained in the previous section. Table 4.2 shows the results of the Shapiro-Wilk’s test for all metrics and experiments. The tests were done using the `shapiro.test` function of R, which receives a single vector of data as parameter and uses the null hypothesis that the distribution of the data in the vector are normally distributed. The function returns the $W$ statistic, and the $p$-value. The closer the statistic $W$ is to 1, the more evidence there is for the normality of the sample. The rejection criterion for this test is that the $p$-value $< \alpha$.

The hypothesis testing algorithms compare two data samples to determine if a property relating both samples (the null hypothesis) is statistically significant. When both of the data samples are normally distributed one of the variations of the $t$-test is adequate to test the hypothesis. If any of the data samples is not normally distributed, it is recommended to use a non-parametric hypothesis testing algorithm. A test is non-parametric if it does not make any assumption about the distribution of the data samples. A good example of
Table 4.2: Results of the normality tests for the system metrics of all experiments using the Shapiro-Wilk's test

<table>
<thead>
<tr>
<th>Metric</th>
<th>Experiment</th>
<th>P-value(^a) ((\alpha = 0.05))</th>
<th>Statistic(^b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>1</td>
<td>0.244</td>
<td>0.9759</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.081</td>
<td>0.9662</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.010</td>
<td>0.9456</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.831</td>
<td>0.9874</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.399</td>
<td>0.9787</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.695</td>
<td>0.9830</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>1</td>
<td>0.243</td>
<td>0.9759</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.153</td>
<td>0.9716</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.778</td>
<td>0.9871</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.902</td>
<td>0.9892</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.081</td>
<td>0.9638</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.531</td>
<td>0.9792</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>1</td>
<td>0.000</td>
<td>0.7720</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.000</td>
<td>0.8728</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.000</td>
<td>0.8289</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.000</td>
<td>0.8078</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.000</td>
<td>0.8727</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.000</td>
<td>0.8284</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>1</td>
<td>0.011</td>
<td>0.9231</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.384</td>
<td>0.9797</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.396</td>
<td>0.9792</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.427</td>
<td>0.9785</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.615</td>
<td>0.9835</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.060</td>
<td>0.9553</td>
</tr>
<tr>
<td>Percentage of Time with at Least One Overloaded Sector (%OL)</td>
<td>1</td>
<td>0.520</td>
<td>0.9829</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.250</td>
<td>0.9758</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.808</td>
<td>0.9877</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.249</td>
<td>0.9730</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.422</td>
<td>0.9793</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.069</td>
<td>0.9567</td>
</tr>
<tr>
<td>Percentage of Time with at Least One Over-Scheduled Airport (%Osch)</td>
<td>1</td>
<td>0.000</td>
<td>0.8991</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.000</td>
<td>0.8908</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.000</td>
<td>0.8490</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.000</td>
<td>0.8821</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.000</td>
<td>0.8623</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.000</td>
<td>0.8923</td>
</tr>
</tbody>
</table>

\(^a\) A sample is not normal if \(p < \alpha\).  \(^b\) Using Shapiro-Wilk's test for normality.
non-parametric hypothesis testing algorithm is the *Wilcoxon* test.

Table 4.2 shows that some of the comparisons required in the design of experiment will use the Wilcoxon test instead of the t-test.

### 4.3 Analyzing the Effects of Availability to Global Information

In the experiment #1 the adaptable AOCs have access to global information to select flightplan routes. As a result, if airline A is selecting a route, it can access data for fuel burn, airborne conflicts, departure delay, arrival delay, distance, and the percentage of congested sectors crossed from all the other airlines, $B_i$, that have flights at about the same time (i.e., ±15 minutes from the scheduled departure), and for the same O/D pair, as airline A (see Table 3.4 on page 88). All the data are available to the airlines without delay and without error (see Table 3.5 on page 95).

In experiment #2 the adaptable AOCs have access to data of their airline for fuel burn, departure delay, arrival delay, and distance. There is no information about the percentage congested sectors crossed nor the airborne conflicts. In this experiment, all the data are available to the airline without delay and without error (see Table 3.5).

The following paragraphs compare the experiments one system metric at a time. The complete evolution of the metrics is shown and the point in time in which the learning process reaches steady-state (see Table 4.1) is indicated in the charts.

Figure 4.2 compares the *total fuel burn* for experiments #1 and #2 from the first simulated day to the day 80. The fuel burn starts at about 300 million kilograms per day of fuel when the AOCs select routes randomly (when $\varepsilon = 1$). With the gradual change from random selection to a greedier selection the fuel burn of the system reduces. At day 17, when $\varepsilon = 0.2$ and the learning process enters the steady-state, the simulation using global information burns 299.6 million kilograms per day of fuel, a reduction of 0.471 million kilograms / day. At day 18, experiment #2 reaches steady-state and burns 299.5 million kilograms /
day, a reduction of 0.664 million kilograms per day.

During the exploitation period, when the learning process is stable, there is a trend toward reduction of the fuel burn, but the slope of the reduction is small compared to the reduction during the exploration period. The total fuel burn for the experiment using global information is 299.57 million kilograms / day, whereas it is 299.39 million kilograms / day for the experiment using local information. This is an increase of 0.1% in total fuel burn with respect to mean of experiment using local information.

In both experiments, there is variation of the fuel burn between days even with a deterministic model of the NAS and without weather effects. These variations are attributed to the interaction of the learning processes of the AOCs. The standard deviation when AOCs have access to global information is 0.081 million kilograms, and it is 0.066 million kilograms when the AOCs only have local information. The reason for the increase in variance is that AOCs using more information must search for better alternatives in a bigger search space; these AOCs must optimize more variables, than the AOC with only local information.

It is plausible that the AOCs trade-off fuel burn when trying to optimize the other variables. This phenomenon can be related to the information overloading described before for Net-Centric Operations: excessive information only makes the decision process more complex, not more effective[Alberts, 1996b].

Figure 4.3 shows the evolution of the total airborne conflicts for experiments #1 and #2. The total airborne conflicts start at 65,072 per day when the AOCs select routes randomly ($\varepsilon = 1$) and have global information, and at 64,642 per day when the AOCs have local information only. With the step-by-step change from random selection to a greedier selection, the total airborne conflicts of the system reduce. At day 17 ($\varepsilon = 0.2$) the simulation using global information has 60,590 conflicts / day, a reduction of 4,482 conflicts per day. At day 18, the experiment #2 has 62,601 conflicts / day, a reduction of 2,041 conflicts / day.

When the learning process becomes stable, there is a trend toward reduction of the total airborne conflicts, but the slope of the reduction is small compared to the reduction during
the exploration period. The average airborne conflicts for the experiment using global information is 60,546 conflicts / day, whereas it is 62,259 conflicts / day for the experiment using local information. This difference of 2.8% with respect to mean of experiment using local information indicates that performance improved when the adaptable AOCs have global information, i.e. the system shows less airborne conflicts. This difference occurs because the AOCs using local information only do not have data about the conflicts, but the AOCs using global information optimize the conflicts for each of their flights.

In both experiments, there is variation of the total airborne conflicts between days despite a deterministic model of the NAS without weather effects. As with fuel burn, these variations can be attributed to the interaction of the learning processes of the AOCs. The standard deviation when AOCs have access to global information is 340, and it is 178 when the AOCs only have local information. The reason for the increase in variance is that AOCs
using more information must search for better alternatives in a bigger search space; these AOCs must optimize more variables, than the AOC with only local information. AOCs with local information do not consider conflicts when trading-off variables, but AOCs with global information must find alternate routes that optimize airborne conflicts, and are before a more complex decision-making process.

Figure 4.3: Total airborne conflicts when the AOCs have local or global, real-time, and accurate information.

Figure 4.4 shows the evolution of the total departure delay for the experiments #1 and #2. The total departure delay start at 125,600 minutes / day when the AOCs select routes randomly ($\varepsilon = 1$). With the step-by-step change from random selection to a greedier selection the total departure delay of the system reduces, but shows significant variation in the process. At day 17 ($\varepsilon = 0.2$) the simulation using global information has 120,145 minutes / day of departure delay, a reduction of 5,463 minutes / day. At day 18, the
experiment #2 has 121,920 minutes / day, a reduction of 3,454 minutes / day.

When the learning process becomes stable, there is no indication of reduction in the total departure delay. The average departure delay for the experiment using global information is 118,944 minutes / day, whereas it is 118,604 minutes / day for the experiment using local information. This difference of 0.1% with respect to mean of experiment using local information indicates that there is no reduction in the departure delays when the AOCs have access to global information.

The standard deviation when AOCs have access to global information is 2,486 minutes / day, and it is 2,927 minutes / day when the AOCs only have local information. The standard deviations are two orders of magnitude smaller than the means, and their values are comparable. In this metric, there is no indication of significant effects of the availability of global information. This occurs because the difference in distance between the alternate routes is not big enough to modify the pattern of arrivals at the destination airports. Hence, the congestion is similar with any alternate route and FACET assigns similar departure delays to the flights. As a result, the averages are similar and the variations are small.

Figure 4.5 shows the evolution of the total arrival delay for the experiments #1 and #2. The total arrival delay start at about 571,238 minutes / day (8.43 minutes / flight) when the AOCs select routes randomly ($\varepsilon = 1$) with global information, and at 570,463 minutes / day (8.42 minutes / flight) with local information only. With the step-by-step change from random selection to a greedier selection the total arrival delay of the system reduces. At day 17 ($\varepsilon = 0.2$) the simulation using global information has 559,818 minutes / day of arrival delay (8.26 minutes / flight), a reduction of 11,420 minutes / day (10.1 second / flight). At day 18, the experiment #2 has 559,955 minutes / day (8.26 minutes / flight), a reduction of 10,508 minutes / day (9.31 seconds / flight).

When the learning process becomes stable, there is a trend toward reduction in the total arrival delay, but the slope of the trend is smaller than the slope in the exploration period. The average arrival delay for the experiment using global information is 557,990 minutes / day (8.23 minutes / flight), whereas it is 556,644 minutes / day (8.21 minutes / flight) for
the experiment using local information. This difference of 0.20% (1 second / flight) with respect to mean of the experiment using local information indicates that there is no effect on the total arrival delays when the AOCs have access to global information.

Figure 4.6 shows the evolution of the %OL for the experiments #1 and #2. The %OL starts at 74.8% when the AOCs select routes randomly (ε = 1) with global information, and at 73.5% with local information only. With the step-by-step change from random selection to a greedier selection the %OL of the system shows little reduction, but significant variation. At day 17 (ε = 0.2) the simulation using global information is at 73.8% of %OL, a reduction of 1% or 14.4 simulated minutes less with at least one sector above its MAP value. At day 18, the experiment #2 is at 73.0% of %OL, a reduction of 0.5% or 7.2 simulated minutes less with at least one sector above its MAP value. A difference of 15 minutes in 24 hours (the total simulated time) is an indication that the use of greedy route selection and adaptation
Figure 4.5: Total arrival delay when the AOCs have local or global, real-time, and accurate information.

is having no effect in the time distribution of sector congestion.

When the learning process becomes stable, there is no indication of reduction in the %OL. The average %OL for the experiment using global information is 73.38%, whereas it is 72.99% for the experiment using local information. This difference of 0.39% (5.6 simulated minutes) indicates that there is no reduction in the %OL when the AOCs have access to global information.

The standard deviations in the exploitation period are both less than 1%. The deviation is 0.464% when the AOCs have global information, and 0.385% when they have local information. This difference is about 1 simulated minute in 24 simulated hours. This is another indication that there is no effect using adaptable route selection and having global information.

Figure 4.7 shows the evolution of the %Osch. This metric does not show any significant
Figure 4.6: %OL when the AOCs have local or global, real-time, and accurate information.

change in the simulated days of the stable period. The reason for the apparent lack of effect of the independent variable is that congestion at the arrival airports is not changed by flightplan route selection, unless the difference in distance of the alternate routes is shifts the arrival time by more than 15 minutes. An AOC would not choose a much longer route to reduce congestion in the arrival airport, because the choice would mean much higher fuel cost and little benefit for the airline. Previous studies have suggested [Calderon-Meza et al., 2009, Calderon-Meza and Sherry, 2010a] that flightplan route selection only shifts in time the congestion at the arrival airports, but it does not reduce or increase it. These results confirm that conclusion.

The next step of the analysis examines the comparison between experiments. First, the goal is to determine if the variances of the metrics in the stable period are equal between experiments. This information helps in the decision of how to compare the means of the samples.
Figure 4.7: %Osch when the AOCs have local or global, real-time, and accurate information.

The F-test used to compare the variances of two data samples were done with the var.test function of statistical software R (version 2.11.1) with the parameters described in Table 4.3. The null hypothesis, $H_0$, in an F-test is defined by the ratio between the two variances of the samples. Hence, if the null hypotheses states that

$$H_0 : \frac{s_1^2}{s_2^2} = 1$$

The test is testing for equality of the variances. Other hypotheses are possible to determine which of the variances is greater and what the value of the ratios should be. In this test, the goal is to determine the equality of the variances; this justifies the use of “two-sided” as alternate hypothesis, and an expected ratio of 1.
Table 4.3: Parameters for the F-test to determine equality of sample variances

<table>
<thead>
<tr>
<th>Argument name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Vector with the first sample population to test</td>
<td>Metrics values for experiment #1.</td>
</tr>
<tr>
<td>Y</td>
<td>Vector with the second sample population to test</td>
<td>Metrics values for experiment #2.</td>
</tr>
<tr>
<td>Ratio</td>
<td>The hypothesized ratio of the population variances of x and y</td>
<td>1</td>
</tr>
<tr>
<td>Alternative</td>
<td>A character string specifying the alternative hypothesis. Allowed values are “two.sided” (default), “greater” or “less”</td>
<td>“two.sided”</td>
</tr>
<tr>
<td>conf.level</td>
<td>Confidence level for the returned confidence interval</td>
<td>0.95</td>
</tr>
</tbody>
</table>

The interpretation of the test results can be done in two ways. First, the given p-value is the smallest value of $\alpha$ for which the $F$-statistic is significantly different from the hypothesized value of 1. Therefore, if the p-value $> \alpha$, the null hypothesis cannot be rejected with $\alpha$ significance. Second, if the value of the F-statistic falls inside of the given confidence interval, the null hypothesis cannot be rejected with $\alpha$ significance. A rejection of the null hypothesis means that the variances of the two samples are significantly different (the alternate hypothesis is true).

Table 4.4 shows that the standard deviations for the metrics are statistically equal between experiments for fuel burn, departure delay, %OL, and %Osch. Only for the total airborne conflicts and arrival delay the standard deviations are significantly different between experiments. These results provide evidence that the variance of the metrics is not affected by the independent variable.

The second goal of the analysis is to compare the means of the samples for each metrics. Table 4.2 indicates that fuel burn, conflicts, and % of time with overloaded sectors are normally distributed in both experiments. Departure delay and %Osch are not normally distributed in both experiments. Arrival delay is an exceptional case in which the distribution is normal for experiment #2, but not normal for experiment #1. In the cases where the distribution is normal in both samples, a parametric test can be used. In this case, the number of samples exceeds 30 and the recommendation is the unpaired t-test. For the cases in which the variances are equal between experiments the unpaired t-test for equal variances
Table 4.4: Tests for equality of variances of the system-wide performance metrics when AOCs have global information vs. local information

<table>
<thead>
<tr>
<th>Metric</th>
<th>P-value(^a) ((\alpha = 0.05))</th>
<th>Ratio</th>
<th>95% CI</th>
<th>Statistic(^b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>0.110</td>
<td>1.503</td>
<td>[0.911, 2.478]</td>
<td>1.503</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>0.000</td>
<td>3.642</td>
<td>[2.207, 6.004]</td>
<td>3.642</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>0.065</td>
<td>N/A</td>
<td>N/A</td>
<td>3.456</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>0.020</td>
<td>N/A</td>
<td>N/A</td>
<td>5.533</td>
</tr>
<tr>
<td>%OL</td>
<td>0.143</td>
<td>1.453</td>
<td>[0.881, 2.396]</td>
<td>1.453</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.671</td>
<td>N/A</td>
<td>N/A</td>
<td>0.182</td>
</tr>
</tbody>
</table>

\(^a\) Null hypothesis rejected if p < \(\alpha\).  \(^b\) Using F-test or Levene’s test for the comparison of variances.

is the best option. For the case of the conflicts, the unpaired t-test for non-equal variances is used. For the cases in which, at least, one distribution is not normal, the comparison is done with the non-parametric Wilcoxon test.

The t-tests were done with the t.test function of R with the parameters described in Table 4.5. The Wilcoxon tests were made with the Wilcox.test function of R. For these two tests, the null hypothesis is that the difference between the means of the two samples equal 0, or that the means of the samples are equal. In this case also the p-value is the smallest value of \(\alpha\) for which the difference between the means is significantly different from the hypothesized value (0 in this case). Therefore, if the p-value < \(\alpha\) then null hypothesis is rejected with \(\alpha\) significance. Another way to interpret the result of the t-test is that if the given confidence interval includes the hypothesized value, the null hypothesis cannot be rejected with \(\alpha\) confidence.

The first null hypotheses for performance, \(H^1_0\), states that the performance when system-wide information is present decreases compared to the performance when only local information is available. All the metrics used in these experiments point to better performance when their values are lower. Hence, a decrease in performance is actually an increase in the metrics values. The null hypothesis can be expressed formally as follows:

\[
H^1_0 : \bar{x}_{global} \geq \bar{x}_{local}
\]
Table 4.5: Parameters of the t-tests to determine differences between the mean values

<table>
<thead>
<tr>
<th>Argument name</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Vector with the first sample population to test</td>
<td>Metrics values for experiment #1</td>
</tr>
<tr>
<td>Y</td>
<td>Vector with the second sample population to test</td>
<td>Metrics values for experiment #2</td>
</tr>
<tr>
<td>alternative</td>
<td>A character string specifying the alternative hypothesis. Allowed values are “two.sided” (default), “greater” or “less”</td>
<td>“less”</td>
</tr>
<tr>
<td>mu</td>
<td>A number indicating the true value of the difference in means</td>
<td>0</td>
</tr>
<tr>
<td>Paired</td>
<td>A logical indicating whether a paired t-test is requested</td>
<td>FALSE</td>
</tr>
<tr>
<td>Var.equal</td>
<td>A logical variable indicating whether to treat the two variances as being equal</td>
<td>Depends on F-test result</td>
</tr>
<tr>
<td>conf.level</td>
<td>Confidence level for the returned confidence interval</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Table 4.6: Hypothesis testing for the means of the system-wide metrics when AOCs have global information vs. local information

<table>
<thead>
<tr>
<th>Metric</th>
<th>P-value(^a) ((\alpha = 0.05))</th>
<th>Difference (%)</th>
<th>95% CI</th>
<th>Statistic(^b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>1.000</td>
<td>174,607 (0.1%)</td>
<td>([-\infty, 196,288])</td>
<td>t = 13.3</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>0.000</td>
<td>-1,713 (2.8%)</td>
<td>([-\infty, -1,633])</td>
<td>t = -35.6</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>0.592</td>
<td>81 (0.1%)</td>
<td>([-\infty, 564])</td>
<td>W = 2,063.5</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>1.000</td>
<td>1,341 (0.2%)</td>
<td>([-\infty, 1,877])</td>
<td>W = 2,873.0</td>
</tr>
<tr>
<td>%OL</td>
<td>1.000</td>
<td>0.40 (0.5%)</td>
<td>([-\infty, 0.5226])</td>
<td>t = 5.241</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.989</td>
<td>0.00 (0.0%)</td>
<td>([-\infty, 0.0099])</td>
<td>W = 2,459.0</td>
</tr>
</tbody>
</table>

\(^a\) Null hypothesis rejected if \(p < \alpha\).  \(^b\) Using unpaired t-test or Wilcoxon’s test for the difference between means.

So, the t and Wilcoxon tests must be one-sided and the alternative hypothesis is that the mean for experiment #1 is smaller than the mean for experiment #2. Hence, the tests should use “less” as alternate hypothesis parameter.

Table 4.6 summarizes the results for the tests comparing the means of the samples. The tests resulting in rejection of the null hypothesis confirm the \(H_0^1\) hypothesis. Only the total airborne conflicts confirm the \(H_0^3\) hypothesis. In the other metrics, the availability of global information could result in decreased system performance.

The next step of the analysis is the comparison of the variation of the metrics. The null
Table 4.7: Hypothesis testing for the variances of the system-wide performance metrics when AOCs have global information vs. local information

<table>
<thead>
<tr>
<th>Metric</th>
<th>P-value a ((\alpha = 0.05))</th>
<th>Ratio (%)</th>
<th>95% CI</th>
<th>Statistic b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>0.945</td>
<td>1.503 (50%)</td>
<td>[0.000, 2.285]</td>
<td>1.434</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>1.000</td>
<td>3.642 (264%)</td>
<td>[0.000, 5.537]</td>
<td>3.642</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>0.100</td>
<td>0.721 (-28%)</td>
<td>[0.000, 1.097]</td>
<td>0.721</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>0.032</td>
<td>0.624 (-38%)</td>
<td>[0.000, 0.949]</td>
<td>0.624</td>
</tr>
<tr>
<td>%OL</td>
<td>0.929</td>
<td>1.453 (45%)</td>
<td>[0.000, 2.209]</td>
<td>1.453</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.562</td>
<td>1.041 (4%)</td>
<td>[0.000, 1.583]</td>
<td>1.041</td>
</tr>
</tbody>
</table>

a Null hypothesis rejected if p < \(\alpha\).  
b Using F-test for the ratio of variances.

The \(H^2_0\) hypothesis, states that the availability of system-wide information for flightplan route selection results increased variation of the performance metrics. More formally,

\[
H^2_0: \frac{s^2_{\text{global}}}{s^2_{\text{local}}} \geq 1
\]

The \(H^2_0\) null hypothesis requires running one-sided F-tests. The standard deviation of experiment #1 must be greater than the standard deviation of experiment #2 for null hypothesis to be accepted. This type of test can be done by using the var.test function of R with the same parameters described in Table 4.3, but using alternative=“less” instead of “two.sided”.

Table 4.7 summarizes the results for the F-test with the null hypothesis that the variance of experiment #1 is greater or equal to the variance of experiment #2. In five metrics there is no evidence to reject \(H^2_0\); the variance of the metrics could increase with the use of system-wide information. Only in the total arrival delay metric the null hypothesis is rejected.
4.4 Analyzing the Effects of Latency with Global Information Available

The comparison of experiments #3 and #1 measures the effect of latency in the information used by the airlines to select flightplan routes.

Figure 4.8 compares the total fuel burn for experiments #3 and #1 from the first simulated day to the day 80. The fuel burn starts at about 300.0 million kilograms per day of fuel when the AOCs select routes randomly ($\varepsilon = 1$). With the step-by-step change from random selection to a greedier selection the total fuel burn reduces. At day 21, in the beginning of the steady-state, the simulation using delayed global information burns 299.6 million kilograms / day of fuel, a reduction of 0.44 million kilograms / day. At day 17, the experiment #1 burns 299.6 million kilograms / day, a reduction of 0.47 million kilograms / day.

During the exploitation period, when the learning process is stable, there is a trend toward a reduction in fuel burn, but the slope of the reduction is small compared to the reduction during the exploration period. The average fuel burn for the experiment using delayed global information is 299.55 million kilograms / day, and it is 299.57 million kilograms / day for the experiment using real-time global information. This difference of 18,099 kilograms / day (0.0%) with respect to the mean of experiment using real-time global information is a first indication that performance remained unchanged when the adaptable AOCs have delayed information.

In both experiments, there is variation of the fuel burn between days with a deterministic model of the NAS without weather effects. These variations can be attributed to the interaction of the learning processes of the AOCs. The standard deviation when AOCs have access to delayed global information is 0.074 million kilograms, is 0.081 million kilograms when the AOCs have real-time global information.

Figure 4.9 shows the evolution of the total airborne conflicts for experiments #3 and #1. The total airborne conflicts start at 64,641 per day when the AOCs select routes randomly
Figure 4.8: Total fuel burn when the AOCs have delayed or real-time global accurate information.

($\epsilon = 1$) and have delayed global information, and at 65,072 per day when the AOCs have real-time global information. With the step-by-step change from random selection to a greedier selection the total airborne conflicts of the system reduce. At the day 21 (i.e. start of steady-state), the simulation using delayed global information has 61,033 conflicts / day, a reduction of 3,608 conflicts per day. At day 17, the experiment #1 has 60,590 conflicts / day, a reduction of 4,482 conflicts / day.

When the learning process becomes stable, there is a trend toward reduction of the total airborne conflicts, but the slope of the reduction is small compared to the reduction during the exploration period. The average airborne conflicts for the experiment using delayed global information is 60,673 conflicts / day, whereas it is 60,546 conflicts / day for the experiment using real-time global information. This difference of 0.2% with respect to mean of experiment using real-time global information indicates that performance decreased when
the adaptable AOCs have delayed global information, i.e. the system shows more airborne conflicts. This difference happens because the AOCs are using outdated information to select routes and their decisions are suboptimal.

In both experiments, there is variation of the total airborne conflicts between days with a deterministic model of the NAS without weather effects. As with fuel burn, these variations can only be attributed to the interaction of the learning processes of the AOCs. The standard deviation when AOCs have access to delayed global information is 271, and it is 340 when the AOCs only have real-time global information.

Figure 4.9: Total airborne conflicts when the AOCs have delayed or real-time global information.

Figure 4.10 shows the evolution of the total departure delay for the experiments #3 and #1. The total departure delay starts at 125,600 minutes / day when the AOCs select routes randomly ($\varepsilon = 1$). With the step-by-step change from random selection to a greedier
selection the total departure delay of the system reduces, but shows significant variation in
the process. At day 21, i.e. the start of steady-state, the simulation using delayed global
information has 121,546 minutes / day of departure delay, a reduction of 4,130 minutes /
day. At day 17, the experiment #1 has 120,145 minutes / day, a reduction of 5,463 minutes /
day.

When the learning process becomes stable, there is no indication of reduction in the
total departure delay. The average departure delay for the experiment using delayed global
information is 121,956 minutes / day, whereas it is 118,944 minutes / day for the experi-
ment using real-time global information. This difference of 2.0% with respect to mean of
experiment using real-time global information indicates that there is an increase in the total
departure delay when the AOCs have access to delayed global information.

The standard deviation when AOCs have access to delayed global information is 1,445
minutes / day, and it is 2,486 minutes / day when the AOCs have real-time global infor-
mation. The standard deviations are two orders of magnitude smaller than the means, and
their values are comparable. A moderate delay (of 1 day) in the information tends to reduce
the variation of the metric.

Figure 4.11 shows the evolution of the total arrival delay for the experiments #3 and
#1. The total arrival delay starts at about 571,238 minutes / day (8.43 minutes / flight)
when the AOCs select routes randomly ($\varepsilon = 1$) with real-time global information, and at
about 570,124 minutes / day (8.41 minutes / flight) with delayed global information only.
With the step-by-step change from random selection to a greedier selection the total arrival
delay of the system reduces. At day 17 (i.e. start of steady-state) the simulation using real-
time global information has 559,818 minutes / day of arrival delay (8.26 minutes / flight),
a reduction of 11,420 minutes / day (10.1 second / flight). At day 21, the experiment using
delayed global information has 560,612 minutes / day (8.27 minutes / flight), a reduction
of 9,512 minutes / day (8.42 seconds / flight).

When the learning process becomes stable, there is a trend toward reduction in the total
arrival delay, but the slope of the trend is smaller than the slope in the exploration period.
Figure 4.10: Total departure delay when the AOCs have delayed or real-time global accurate information.

The average arrival delay for the experiment using real-time global information is 557,990 minutes / day (8.23 minutes / flight), whereas it is 559,797 minutes / day (8.26 minutes / flight) for the experiment using delayed global information. This difference of 0.3% (1 second / flight) with respect to mean of experiment using real-time global information indicates that there is no effect on the total arrival delays when the AOCs have access to delayed global information.

The standard deviation when AOCs have access to delayed global information is 1,419 minutes / day, and it is 1,545 minutes / day when the AOCs have real-time global information. The standard deviations are two orders of magnitude smaller than the means, and their values are comparable. A moderate delay (of 1 day) in the information tends to reduce the variation of the metric.

Figure 4.12 shows the evolution of the %OL for the experiments #3 and #1. The
Figure 4.11: Total arrival delay when the AOCs have delayed or real-time global accurate information.

%OL starts at 74.8% when the AOCs select routes randomly (ε = 1). With the step-by-step change from random selection to a greedier selection the %OL of the system shows little reduction, but significant variation. At day 17, the simulation using real-time global information is at 73.8% of %OL, a reduction of 1% or 14.4 simulated minutes less with at least one sector above its MAP value. At day 21, the experiment using delayed global information is at 73.5% of %OL, a reduction of 0.3% or 4.3 simulated minutes less with at least one sector above its MAP value. A difference of 10 minutes in 24 hours (the total simulated time) is an indication that the use of greedy route selection and adaptation is having no effect in the time distribution of sector congestion even in the case when the information is delayed.

When the learning process becomes stable, there is no indication of reduction in the %OL. The average %OL for the experiment using real-time global information is 73.38%,
whereas it is 73.20% for the experiment using delayed global information. This difference of 0.18% (2.6 simulated minutes) indicates that there is no reduction in the %OL when the AOCs have access to delayed global information.

The standard deviations in the exploitation period are both less than 1%. The deviation is 0.464% when the AOCs have real-time global information, and 0.439% when they have delayed global information. This difference is about 21 seconds of simulated time in 24 simulated hours, which is another indication that there is no effect using adaptable route selection and having delayed global information.

Figure 4.12: %OL when the AOCs have delayed or real-time global accurate information.

Figure 4.13 shows the constant value of the %Osch metric during the entire experiment. The reason this apparent lack of effect of the independent variable is that congestion at the arrival airports is not changed by flightplan route selection, even when the information is delayed.
Figure 4.13: %Osch when the AOCs have delayed or real-time global accurate information.

The analysis of the results for experiments #1 and #3 follows the process described in section 3.6 on page 97.

Table 4.8 shows that the variances for the metrics are different between experiments for total airborne conflicts, total departure delay, and total arrival delay. However, no significant difference exists between the standard deviations of the total fuel burn, %OL, and %Osch. These results provide evidence of the statistically significant effect of the independent variable on three of the system performance metrics.

The comparison of the means also follows the same methodology described in the previous section. However, the null hypothesis, $H_0^3$, states that the presence of latency results in increased or equal performance of the system. More formally,

$$H_0^3 : \bar{x}_{\text{latency}} \leq \bar{x}_{\text{realtime}}$$
Table 4.8: Tests for equality of variances of the system-wide performance metrics when AOCs have delayed information vs. real-time information

<table>
<thead>
<tr>
<th>Metric</th>
<th>P-value&lt;sup&gt;a&lt;/sup&gt; (α = 0.05)</th>
<th>Ratio</th>
<th>95% CI</th>
<th>Statistic&lt;sup&gt;b&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>0.534</td>
<td>0.851</td>
<td>[0.514, 1.417]</td>
<td>0.851</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>0.081</td>
<td>0.636</td>
<td>[0.384, 1.057]</td>
<td>0.636</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>0.024</td>
<td>N/A</td>
<td>N/A</td>
<td>5.202</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>0.780</td>
<td>N/A</td>
<td>N/A</td>
<td>0.078</td>
</tr>
<tr>
<td>%OL</td>
<td>0.667</td>
<td>0.895</td>
<td>[0.540, 1.488]</td>
<td>0.895</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.847</td>
<td>N/A</td>
<td>N/A</td>
<td>0.037</td>
</tr>
</tbody>
</table>

<sup>a</sup> Null hypothesis rejected if p < α.  <sup>b</sup> Using F-test or Levene’s test for the comparison of variances.

Table 4.9: Hypothesis testing for the means of the system-wide metrics when AOCs have global delayed information vs. global real-time information

<table>
<thead>
<tr>
<th>Metric</th>
<th>P-value&lt;sup&gt;a&lt;/sup&gt; (α = 0.05)</th>
<th>Difference (%)</th>
<th>95% CI</th>
<th>Statistic&lt;sup&gt;b&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>0.875</td>
<td>-18,099 (0.0%)</td>
<td>[-42,287, ∞]</td>
<td>t = 1,690.0</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>0.012</td>
<td>127 (0.2%)</td>
<td>[35, ∞]</td>
<td>t = 2.3</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>0.000</td>
<td>2,384 (2.0%)</td>
<td>[1,971, ∞]</td>
<td>W = 3,587.0</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>0.000</td>
<td>1,698 (0.3%)</td>
<td>[1,279, ∞]</td>
<td>t = 3,166.0</td>
</tr>
<tr>
<td>%OL</td>
<td>0.988</td>
<td>-0.19% (0.19%)</td>
<td>[-0.3209, ∞]</td>
<td>t = -2.291</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.000</td>
<td>0.01% (0.01%)</td>
<td>[0.0099, ∞] %</td>
<td>W = 3,073</td>
</tr>
</tbody>
</table>

<sup>a</sup> Null hypothesis rejected if p < α.  <sup>b</sup> Using unpaired t-test or Wilcoxon’s test for the difference between means.

The parameters described in Table 4.5 are valid for this comparison. Nevertheless, the alternative hypotheses must be that the mean for experiment #3 is greater than the mean for experiment #1. The parameter for the test function must be “greater” if experiment #3 is the first dataset into the function.

Table 4.9 shows that the null hypothesis, $H^3_{0}$, is rejected with a 95% confidence for all the system metrics except for %OL. These results imply that in most cases, the presence of latency in the information used for flightplan selection decreases performance (i.e., greater values for the mean of the metrics).

The null hypothesis, $H^4_{0}$, states that the presence of latency results in decreased or equal
Table 4.10: Hypothesis testing for the variance of the system-wide performance metrics when AOCs have global delayed information vs. global real-time information

<table>
<thead>
<tr>
<th>Metric</th>
<th>P-value(^a) ((\alpha = 0.05))</th>
<th>Ratio (%)</th>
<th>95% CI</th>
<th>Statistic(^b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>0.733</td>
<td>0.851 (-15%)</td>
<td>[0.558, (\infty)]</td>
<td>0.766</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>0.960</td>
<td>0.636 (-36%)</td>
<td>[0.416, (\infty)]</td>
<td>0.636</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>1.000</td>
<td>0.338 (-66%)</td>
<td>[0.221, (\infty)]</td>
<td>0.338</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>0.745</td>
<td>0.843 (-16%)</td>
<td>[0.552, (\infty)]</td>
<td>0.843</td>
</tr>
<tr>
<td>%OL</td>
<td>0.666</td>
<td>0.895 (-11%)</td>
<td>[0.586, (\infty)]</td>
<td>0.895</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.904</td>
<td>0.713 (-29%)</td>
<td>[0.467, (\infty)]</td>
<td>0.713</td>
</tr>
</tbody>
</table>

\(a\) Null hypothesis rejected if \(p < \alpha\).  \(b\) Using F-test for the ratio of variances.

variance in the system performance. More formally,

\[ H_0^4: \frac{s^2_{\text{latency}}}{s^2_{\text{realtime}}} \leq 1 \]

The parameters for the test are as described in Table 4.3 on page 115, but the alternate hypothesis parameter must be “greater” in this case if the invocation of the test function uses the dataset of experiment #3 as first parameter and the dataset for experiment #1 as second.

Table 4.10 shows that the null hypothesis, \(H_0^4\), cannot be rejected for any of the metrics with 95% confidence. This result implies that the variation of the metrics could decrease in the presence of latency in the SWIM data used for flightplan route selection.

### 4.5 Analyzing the Effects of Latency with Local Information Available

The comparison of experiments #4 and #2 measures the effect of latency in the information used by the AOCs to select flightplan routes. In this scenario, the AOCs have access to local information and the information is accurate, but in experiment #4 the information is delayed one day.
Figure 4.14 compares the total fuel burn for experiments #4 and #2 from the first simulated day to the day 80. The fuel burn starts at about 300.1 million kilograms per day of fuel when the AOCs select routes randomly ($\varepsilon = 1$). With the gradual change from random selection to a greedier selection the total fuel burn reduces. At day 26 (i.e. start of steady-state) the simulation using delayed local information burns of 299.4 million kilograms / day of fuel for a reduction of 0.753 million kilograms / day. At day 18, the experiment using real-time local information burn 299.5 million kilograms / day of fuel for a reduction of 0.664 million kilograms / day.

During the exploitation period, when the learning process is stable, there is a trend toward reduction of the fuel burn, but the slope of the reduction is small compared to the reduction during the exploration period. The average fuel burn for the experiment using delayed local information is 299.38 million kilograms / day, and it is 299.39 million kilograms / day for the experiment using real-time local information. This difference of 9,553 kilograms / day (0.0%) with respect to the mean of experiment using real-time local information is a first indication that performance remains unchanged when the adaptable AOCs have delayed local information.

In both experiments, there is variation of the fuel burn between days even with a deterministic model of the NAS and without weather effects. These variations can only be attributed to the interaction of the learning processes of the AOCs. The standard deviation for the experiment using delayed information is 0.069 million kilograms a day. For the experiment using real-time local information the standard deviation is 0.066 million kilograms.

Figure 4.15 shows that the total airborne conflicts start at 64,930 per day when the AOCs select routes randomly ($\varepsilon = 1$) and have delayed local information. The fuel burn starts at 64,642 per day when the AOCs have real-time local information. With the step-by-step change from random selection to a greedier selection the total airborne conflicts of the system reduce. At day 26 (i.e. start of steady-state) the simulation using delayed local information has 62,344 conflicts / day, a reduction of 2,586 conflicts per day. At day 18,
the experiment #2 has 62,601 conflicts / day, a reduction of 2,041 conflicts / day.

When the learning process becomes stable, there is a trend toward reduction of the total airborne conflicts, but the slope of the reduction is small compared to the reduction during the exploration period. The average airborne conflicts for the experiment using delayed local information is 62,668 conflicts / day, whereas it is 62,259 conflicts / day for the experiment using real-time local information. This difference of 0.7% with respect to mean of experiment using real-time local information indicates that performance decreased when the adaptable AOCs have delayed local information, i.e. the system shows more airborne conflicts. This difference happens because the AOCs are using outdated information to select routes, their decision are suboptimal. Furthermore, AOCs do not have information about the conflicts their flights are involved in because of the local-only information. As a result, the AOCs do not optimize their route selections for conflicts.
In both experiments, there is variation of the total airborne conflicts between days even with a deterministic model of the NAS without weather effects. As with fuel burn, these variations can only be attributed to the interaction of the learning processes of the AOCs. The standard deviation when AOCs have access to delayed local information is 209, and it is 178 when the AOCs only have real-time local information. There is no indication of an effect of the latency in the variance of the metric when the AOCs have access to local information.

Figure 4.15: Total airborne conflicts when the AOCs have delayed or real-time local accurate information.

Figure 4.16 shows that the total departure delay starts at 125,374 minutes / day when the AOCs select routes randomly ($\varepsilon = 1$) and use real-time local information. When the AOCs use delayed local information, the total departure delay is 124,311 minutes / day when the selection are done randomly. With the step-by-step change from random selection
to a greedier selection the total departure delay of the system reduces, but shows significant variation in the process. At day 26 (i.e. start of steady-state) the simulation using delayed local information has 120,875 minutes / day of departure delay, a reduction of 3,436 minutes / day. At day 18, the simulation using real-time local information has 121,920 minutes / day, a reduction of 3,454 minutes / day.

When the learning process becomes stable, there is no indication of reduction in the total departure delay. The average departure delay for the experiment using delayed local information is 120,211 minutes / day, whereas it is 119,792 minutes / day for the experiment using real-time local information. This difference of 0.5% with respect to mean of experiment using real-time local information indicates that there is an increase in the total departure delay when the AOCs have access to delayed local information.

The standard deviation when AOCs have access to delayed local information is 1,198 minutes / day, and it is 2,927 minutes / day when the AOCs have real-time local information. The standard deviations are two orders of magnitude smaller than the means, which indicates that the use local information does not generate high variance in the performance metrics. The value for one experiment is about half the value for the other experiment. The difference is suggests that a moderate delay (of 1 day) in the information tends to reduce the variation of the metric.

Figure 4.17 shows that the total arrival delay starts at about 569,525 minutes / day (8.40 minutes / flight) when the AOCs select routes randomly (ε = 1) with delayed local information. When the AOCs use real-time local information the total arrival delay starts at about 570,463 minutes / day (8.41 minutes / flight) with real-time local information only. With the step-by-step change from random selection to a greedier selection the total arrival delay of the system reduces. At day 26 (i.e. start of steady-state) the simulation using delayed local information has 558,852 minutes / day of arrival delay (8.24 minutes / flight), a reduction of 10,673 minutes / day (9.45 second / flight). At day 18, the experiment using real-time local information has 559,955 minutes / day (8.26 minutes / flight), a reduction of 10,508 minutes / day (9.31 seconds / flight).
When the learning process becomes stable, there is a trend toward reduction in the total arrival delay, but the slope of the trend is smaller than the slope in the exploration period. The average arrival delay for the experiment using delayed local information is 557,355 minutes / day (8.22 minutes / flight), whereas it is 556,644 minutes / day (8.21 minutes / flight) for the experiment using real-time local information. This difference of 0.1% (1 second / flight) with respect to mean of experiment using real-time local information indicates that there is no effect on the total arrival delays when the AOCs have access to delayed global information.

The standard deviation when AOCs have access to delayed local information is 1,253 minutes / day, and it is 1,956 minutes / day when the AOCs have real-time local information. The standard deviations are two orders of magnitude smaller than the means, and their values are marginally different. This small difference suggests that a moderate delay in the
information tends to reduce the variation of the total arrival delay.

Figure 4.17: Total arrival delay when the AOCs have delayed or real-time local accurate information.

Figure 4.18 shows that the %OL starts at 73.8% when the AOCs select routes randomly ($\varepsilon = 1$) and have access to delayed local information. When the AOCs have access to real-time information and select routes randomly the %OL is 73.5% in the first simulated day. With the step-by-step change from random selection to a greedier selection the %OL of the system shows little reduction, but significant variation. At day 18 the simulation using real-time local information is at 72.5% of %OL, a reduction of 1% or 14.4 simulated minutes less with at least one sector above its MAP value. At day 26, the experiment using delayed local information is at 73.8% of %OL, a reduction of 0.3% or 4.3 simulated minutes less with at least one sector above its MAP value. A difference of 10 minutes in 24 hours is an indication that the use of greedy route selection and adaptation is having no effect.
in the time distribution of sector congestion even in the case when the information is local and delayed.

When the learning process becomes stable, there is no indication of reduction in the %OL. The average %OL for the experiment using real-time local information is 73.99%, whereas it is 73.36% for the experiment using delayed local information. This difference of 0.63% (8.8 simulated minutes) indicates that there is no reduction in the %OL when the AOCs have access to delayed local information.

The standard deviations in the exploitation period are both less than 1%. The deviation is 0.385% when the AOCs have real-time local information, and 0.414% when they have delayed local information. This difference is about 21 seconds of simulated time in 24 simulated hours, which is another indication that there is no effect using adaptable route selection and having delayed local information.

Figure 4.18: %OL when the AOCs have delayed or real-time local accurate information.
Figure 4.19 indicates that the %Osch does not change significantly during the stable period. The reason this apparent lack of effect of the independent variable is that congestion at the arrival airports is not changed by flightplan route selection, even when the information is local and delayed.

![Figure 4.19: %Osch when the AOCs have delayed or real-time local accurate information.](image)

The comparison of experiments #4 and #2 is the same used to compare experiments #3 and #1. If the datasets are fed to the functions in the order #4 first, and #2 second, all the parameters for the tests are identical. The null hypotheses in this comparison are also $H_{03}^0$ and $H_{04}^0$.

The standard deviations for the metrics are different between experiments for departure and arrival delays as shown in Table 4.11. However, there is no significant difference for fuel burn, conflicts, %OL, and %Osch. These results provide evidence that there is a significant effect of the independent variable on four of the system performance metrics.
Table 4.11: Tests for equality of variances of the system-wide performance metrics when AOCs have local delayed information vs. local real-time information

<table>
<thead>
<tr>
<th>Metric</th>
<th>P-value $^a$ $^{(\alpha = 0.05)}$</th>
<th>Ratio</th>
<th>95% CI</th>
<th>Statistic $^b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>0.716</td>
<td>1.099</td>
<td>[0.656, 1.860]</td>
<td>1.099</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>0.228</td>
<td>1.372</td>
<td>[0.819, 2.321]</td>
<td>1.372</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>0.000</td>
<td>N/A</td>
<td>N/A</td>
<td>27.735</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>0.001</td>
<td>0.410</td>
<td>[0.245, 0.694]</td>
<td>0.410</td>
</tr>
<tr>
<td>%OL</td>
<td>0.580</td>
<td>1.156</td>
<td>[0.690, 1.955]</td>
<td>1.156</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.742</td>
<td>N/A</td>
<td>N/A</td>
<td>0.109</td>
</tr>
</tbody>
</table>

$^a$ Null hypothesis rejected if $p < \alpha$.  $^b$ Using F-test or Levene’s test for the comparison of variances.

Table 4.12: Hypothesis testing for the means of the system-wide metrics when AOCs have local delayed information vs. local real-time information

<table>
<thead>
<tr>
<th>Metric</th>
<th>P-value $^a$ $^{(\alpha = 0.05)}$</th>
<th>Difference (%)</th>
<th>95% CI</th>
<th>Statistic $^b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>0.778</td>
<td>-9,553 (0.0%)</td>
<td>[-30,159, ∞]</td>
<td>$t = -0.800$</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>0.000</td>
<td>409 (0.7%)</td>
<td>[350, ∞]</td>
<td>$t = 11.500$</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>0.016</td>
<td>642 (0.5%)</td>
<td>[143, ∞]</td>
<td>$W = 2,132.5$</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>0.010</td>
<td>711 (0.1%)</td>
<td>[215, ∞]</td>
<td>$t = 2.400$</td>
</tr>
<tr>
<td>%OL</td>
<td>0.000</td>
<td>0.37 (0.4%)</td>
<td>[0.01, ∞]</td>
<td>$t = 5.010$</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.000</td>
<td>0.01 (0.0%)</td>
<td>[0.01, ∞]</td>
<td>$W = 2,798$</td>
</tr>
</tbody>
</table>

$^a$ Null hypothesis rejected if $p < \alpha$.  $^b$ Using unpaired t-test or Wilcoxon’s test for the difference between means.

Table 4.12 shows that the null hypothesis, $H_0^3$, is rejected, with 95% confidence, for all the metrics in this comparison. Thus, the presence of latency in the information used to select flightplan routes causes increases in the averages of all metrics when the AOCs only have access to local information.

Table 4.13 shows that the $H_0^4$ hypothesis cannot be rejected with a 95% confidence for any of the metrics. The presence of latency in the data used to select flightplan routes could result in decreased or equal variance in the system metrics when the airlines have only access to local information.
Table 4.13: Hypothesis testing for the variance of the system-wide performance metrics when AOCs have local delayed information vs. local real-time information

<table>
<thead>
<tr>
<th>Metric</th>
<th>P-value(^a) ((\alpha = 0.05))</th>
<th>Ratio (%)</th>
<th>95% CI</th>
<th>Statistic(^b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>0.358</td>
<td>1.099 (9%)</td>
<td>[0.713, ∞]</td>
<td>1.099</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>0.114</td>
<td>1.372 (37%)</td>
<td>[0.890, ∞]</td>
<td>1.372</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>1.000</td>
<td>0.168 (-83%)</td>
<td>[0.109, ∞]</td>
<td>0.168</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>0.999</td>
<td>0.410 (-59%)</td>
<td>[0.266, ∞]</td>
<td>0.410</td>
</tr>
<tr>
<td>%OL</td>
<td>0.290</td>
<td>1.156 (16%)</td>
<td>[0.750, ∞]</td>
<td>1.156</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.626</td>
<td>0.917 (-8%)</td>
<td>[0.595, ∞]</td>
<td>0.917</td>
</tr>
</tbody>
</table>

\(^a\) Null hypothesis rejected if \(p < \alpha\).  \(^b\) Using F-test for the ratio of variances.

4.6 Analyzing the Effects of Inaccuracies in Global Information

The comparison of experiments #5 and #1 determines the effect of inaccuracies in the global real-time information used by the AOCs to select flightplan routes.

The inaccuracies in the information are generated by superposing a zero-mean normal distribution, \(N(0, \sigma)\), on the measured value. The magnitude of the standard deviation, \(\sigma\), is a parameter of the simulation. In these experiments 30% of the measured value equals \(3\sigma\). The actual magnitude of the inaccuracy is different for each metric, but all of the metrics show inaccuracies with the same distribution and parameters. The same type of inaccuracy is present in the historic data used during the decision-making process.

Figure 4.20 compares the total fuel burn for experiments #5 and #1 from the first simulated day to the day 80\(^{th}\). The fuel burn starts at about 301.1 million kilograms per day of fuel when the AOCs select routes randomly (\(\varepsilon = 1\)) and use noisy information. When the information is accurate, the initial value of fuel burn is 300.1 million kilograms / day. With the step-by-step change from random selection to a greedier selection the total fuel burn decreases. At day 23, the simulation using noisy global real-time information reaches steady-state and burns 300.00 million kilograms / day, a reduction of 0.140 million kilograms / day. At day 17, the simulation using accurate global real-time information burns 299.6...
million kilograms / day, a reduction of 0.471 million kilograms / day.

During the exploitation period, when the learning process is stable, there is a trend toward reduction of the fuel burn, but the slope of the reduction is small compared to the reduction during the exploration period. The average fuel burn for the experiment using noisy global real-time information is 299.98 million kilograms / day, and it is 299.57 million kilograms / day for the experiment using accurate real-time global information. This difference of 410,362 kilograms / day (0.1%) with respect to the mean of experiment using accurate information is a first indication that performance decreases when the adaptable AOCs have noisy global real-time information.

In both experiments, there is variation of the fuel burn between days with a deterministic model of the NAS without weather effects. These variations can be attributed to the interaction of the learning processes of the AOCs. The standard deviation for experiment using noisy information is 0.077 million kilograms per day. The standard deviation for the experiment using accurate information is 0.081 million kilograms.

Figure 4.21 shows that the total airborne conflicts start at 64,563 per day when the AOCs select routes randomly \((\varepsilon = 1)\) and have noisy global real-time information, and at 65,072 per day when the AOCs have accurate real-time global information. With the step-by-step change from random selection to a greedier selection the total airborne conflicts of the system reduce. At day 23 (i.e. start of steady-state) the simulation using noisy information has 59,403 conflicts / day, a reduction of 5,160 conflicts per day. At day 17, the experiment #1 has 60,590 conflicts / day, a reduction of 4,482 conflicts / day.

When the learning process becomes stable, there is a trend toward reduction of the total airborne conflicts, but the slope of the reduction is small compared to the reduction during the exploration period. The average airborne conflicts for the experiment using noisy information is 58,299 conflicts / day, whereas it is 60,546 conflicts / day for the experiment using accurate information. This difference of 3.7% with respect to the mean of the experiment using accurate local information indicates that performance improved when the adaptable AOCs have noisy global real-time information, i.e. the system shows less
Figure 4.20: Total fuel burn when the AOCs have accurate or noisy global real-time information.

In both experiments, there is variation of the total airborne conflicts between days even with a deterministic model of the NAS and without weather effects. As with fuel burn, these variations can only be attributed to the interaction of the learning processes of the AOCs. The standard deviation when AOCs have access to noisy global real-time information is 756, and it is 340 when the AOCs only have accurate information. Inaccuracies in the information tend to make predictability harder.

Figure 4.22 shows that the total departure delay starts at 124,425 minutes / day when the AOCs select routes randomly ($\varepsilon = 1$) and use noisy global real-time information. When the AOCs use accurate real-time global information, the total departure delay starts at 125,608 minutes / day when the selections are done randomly. With the step-by-step change from random selection to a greedier selection the total departure delay of the system reduces, but
Figure 4.21: Total airborne conflicts when the AOCs have accurate or noisy global real-time information.

shows significant variation in the process. At day 23, the simulation using noisy information has 121,034 minutes / day of departure delay, a reduction of 3,391 minutes / day. At day 17, the simulation using accurate information has 120,145 minutes / day, a reduction of 5,463 minutes / day.

When the learning process becomes stable, there is no indication of reduction in the total departure delay. The average departure delay for the experiment using noisy information is 119,652 minutes / day, whereas it is 118,944 minutes / day for the experiment using accurate information. This difference of 0.0% with respect to mean of experiment using accurate information indicates that there no effect on the total departure delay when the AOCs have access to noisy global real-time information.

The standard deviation when AOCs have access to noisy information is 1,312 minutes / day, and it is 2,486 minutes / day when the AOCs have accurate information. The
standard deviations are two orders of magnitude smaller than the means, but their values are different. The variance of the metric reduces when the AOCs have noisy global real-time information.

Figure 4.22: Total departure delay when the AOCs have accurate or noisy global real-time information.

Figure 4.23 shows that the total arrival delay starts at about 569,638 minutes / day (8.40 minutes / flight) when the AOCs select routes randomly ($\epsilon = 1$) with noisy global real-time information, and at about 571,238 minutes / day (8.43 minutes / flight) with accurate global real-time information. With the step-by-step change from random selection to a greedier selection the total arrival delay of the system reduces. At day 23, the simulation using noisy information has 556,532 minutes / day of arrival delay (8.21 minutes / flight), a reduction of 13,106 minutes / day (11.61 second / flight). At day 17, the experiment using accurate information has 559,818 minutes / day (8.26 minutes / flight), a reduction of 11,420 minutes
When the learning process becomes stable, there is a trend toward the reduction in the total arrival delay, but the slope of the trend is smaller than the slope in the exploration period. The average arrival delay for the experiment using noisy information is 552,017 minutes / day (8.14 minutes / flight), whereas it is 558,330 minutes / day (8.24 minutes / flight) for the experiment using accurate information. This difference of 1.0% (5.8 second / flight) with respect to the mean of the experiment using accurate local information indicates that there is no effect on the total arrival delays when the AOCs have access to noisy global real-time information.

The standard deviation when AOCs have access to noisy global real-time information is 2,203 minutes / day, and it is 1,545 minutes / day when the AOCs have accurate information. The standard deviations are two orders of magnitude smaller than the means, but their values are different. This results imply that inaccuracies in the information tend to increase the variance, making predictability harder.

Figure 4.24 shows that the %OL starts at 75.0% when the AOCs select routes randomly ($\epsilon = 1$) and have access to noisy global real-time information. When the AOCs have access to accurate global real-time information and select routes randomly the %OL is 74.9% in the first simulated day. With the step-by-step change from random selection to a greedier selection the %OL of the system shows little reduction, but significant variation. At day 23, the simulation using noisy information is at 73.1% of %OL, a reduction of 1.9% or 27.4 simulated minutes less with at least one sector above its MAP value. At day 17, the experiment using accurate information is at 73.8% of %OL, a reduction of 1.1% or 15.8 simulated minutes less with at least one sector above its MAP value. A difference of 11.6 minutes in 24 hours is an indication that the use of greedy route selection and adaptation is having a small effect in the time distribution of sector congestion even in the case when the information is noisy, global, and real-time.

When the learning process becomes stable, there is no indication of reduction in the %OL. The average %OL for the experiment using noisy global real-time information is
Figure 4.23: Total arrival delay when the AOCs have accurate or noisy global real-time information.

72.17%, whereas it is 73.38% for the experiment using accurate information. This difference of 1.22% (17.6 simulated minutes) indicates that there is a reduction in the %OL when the AOCs have access to noisy global real-time information.

The standard deviations in the exploitation period are both less than 1%. The deviation is 0.299% when the AOCs have noisy information, and 0.464% when they have accurate information. This difference is about 2.4 minutes of simulated time in 24 simulated hours, which is an indication of a small reduction in variance when adaptable AOCs use noisy global real-time information for route selection.

Figure 4.25 suggests that the %Osch does not change significantly during the experiment. The reason this apparent lack of effect of the independent variable is that congestion at the arrival airports is not changed by flightplan route selection, even when the information is noisy, global, and real-time.
The comparison of experiments #5 and #1 uses the process explained in section 3.6 on page 97. All the test parameters are the same used for previous comparisons if the datasets are input to the functions as experiment #5 first, and experiment #1 second. The null hypotheses in this comparison are $H^5_0$ for the difference of the means and $H^6_0$ for the difference of the variances.

Table 4.14 shows that no significant difference exists between the variances in any metric. These results provide evidence that, in terms of predictability, there is no significant effect of the independent variable.

Table 4.15 suggests that inaccuracies in the global information significantly affects the total fuel burn, total departure delay and %Osch. The other three metrics show no statistically significant differences between experiments.

Table 4.16 shows that the null hypothesis $H^6_0$ cannot be rejected with 95% confidence for the metrics. As a result, the predictability of the system performance is apparently not
Table 4.14: Tests for equality of variances of the system-wide performance metrics when AOCs have global inaccurate information vs. global accurate information

<table>
<thead>
<tr>
<th>Metric</th>
<th>P-value&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Ratio</th>
<th>95% CI</th>
<th>Statistic&lt;sup&gt;b&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>0.692</td>
<td>0.901</td>
<td>[0.542, 1.508]</td>
<td>0.901</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>0.000</td>
<td>4.940</td>
<td>[2.972, 8.267]</td>
<td>4.940</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>0.015</td>
<td>N/A</td>
<td>N/A</td>
<td>6.130</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>0.001</td>
<td>N/A</td>
<td>N/A</td>
<td>10.792</td>
</tr>
<tr>
<td>%OL</td>
<td>0.001</td>
<td>0.418</td>
<td>[0.252, 0.700]</td>
<td>0.418</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.512</td>
<td>N/A</td>
<td>N/A</td>
<td>0.433</td>
</tr>
</tbody>
</table>

<sup>a</sup> Null hypothesis rejected if $p < \alpha$.  <sup>b</sup> Using F-test or Levene’s test for the comparison of variances.

Table 4.15: Hypothesis testing for the means of the system-wide metrics when AOCs have global inaccurate information vs. global accurate information

<table>
<thead>
<tr>
<th>Metric</th>
<th>P-value&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Difference (%)</th>
<th>95% CI</th>
<th>Statistic&lt;sup&gt;b&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>0.000</td>
<td>410,362 (0.1%)</td>
<td>[386,680, ∞]</td>
<td>$t = 28.700$</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>1.000</td>
<td>-2,247 (3.7%)</td>
<td>[-2,427, ∞]</td>
<td>$t = -20.800$</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>0.571</td>
<td>-44 (0.0%)</td>
<td>[-473, ∞]</td>
<td>$W = 1,821.5$</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>1.000</td>
<td>-5,857 (1.0%)</td>
<td>[-6,555, ∞]</td>
<td>$t = -70.000$</td>
</tr>
<tr>
<td>%OL</td>
<td>1.000</td>
<td>-1.22 (1.2%)</td>
<td>[-1.33, ∞]</td>
<td>$t = -17.404$</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.759</td>
<td>0.00 (0.0%)</td>
<td>[0.00, ∞]</td>
<td>$W = 1,759$</td>
</tr>
</tbody>
</table>

<sup>a</sup> Null hypothesis rejected if $p < \alpha$.  <sup>b</sup> Using unpaired t-test or Wilcoxon’s test for the difference between means.
4.7 Analyzing the Effects of Inaccuracies in Local Information

The comparison of experiments #6 and #2 analyzes the effect of inaccuracies in the local real-time information used by the AOCs to select flightplan routes. The explanation given in section 4.6 on page 138 for the type and parameters of the inaccuracies is valid for this comparison too.

Figure 4.26 shows that the fuel burn is 299.96 million kilograms / day when the AOCs select routes randomly ($\varepsilon = 1$) and use noisy information. When the information is accurate, the initial value of fuel burn is 300.13 million kilograms / day. With the step-by-step change from random selection to a greedier selection the total fuel burn reduces. At day 32,
Table 4.16: Hypothesis testing for the variance of the system-wide performance metrics when AOCs have global inaccurate information vs. global accurate information

<table>
<thead>
<tr>
<th>Metric</th>
<th>P-value$^a$ ($\alpha = 0.05$)</th>
<th>Ratio (%)</th>
<th>95% CI</th>
<th>Statistic$^b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>0.654</td>
<td>0.901 (10%)</td>
<td>[0.589, $\infty$]</td>
<td>0.901</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>0.000</td>
<td>4.940 (394%)</td>
<td>[3.227, $\infty$]</td>
<td>4.940</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>1.000</td>
<td>0.279 (-72%)</td>
<td>[0.182, $\infty$]</td>
<td>0.279</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>0.003</td>
<td>2.032 (103%)</td>
<td>[1.327, $\infty$]</td>
<td>2.032</td>
</tr>
<tr>
<td>%OL</td>
<td>1.000</td>
<td>0.414 (-59%)</td>
<td>[0.271, $\infty$]</td>
<td>0.414</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.715</td>
<td>0.862 (-14%)</td>
<td>[0.564, $\infty$]</td>
<td>0.862</td>
</tr>
</tbody>
</table>

$^a$Null hypothesis rejected if $p < \alpha$.  
$^b$Using F-test for the ratio of variances.

the simulation using inaccurate local real-time information reaches steady-state and burns 300.04 million kilograms / day, an increase of 0.085 million kilograms / day. At day 18, the simulation using accurate local real-time information burns 299.46 million kilograms / day, a reduction of 0.664 million kilograms / day.

During the exploitation period, when the learning process is stable, there is a trend toward reduction of the fuel burn, but the slope of the reduction is small compared to the reduction during the exploration period. The average fuel burn for the experiment using noisy local real-time information is 299.70 million kilograms / day, and it is 299.27 million kilograms / day for the experiment using accurate real-time local information. This difference of 506,895 kilograms / day (0.2%) with respect to the mean of experiment using accurate information is a first indication that performance decreases marginally when the adaptable AOCs have noisy local real-time information.

In both experiments, there is variation of the fuel burn between days, which can only be attributed to the interaction of the learning processes of the AOCs. The standard deviation is 0.086 million kilograms per day when the AOCs use inaccurate local information, but it is 0.066 million kilograms / day when they use accurate local information. This increase of 70.9% is an indication that the variance increases when the information is noisy.

Figure 4.27 shows that the total airborne conflicts are 64,879 / day when the AOCs
select routes randomly ($\varepsilon = 1$) and have inaccurate local real-time information. When the AOCs have accurate real-time local information the conflicts start at 64,642 / day. With the progressive change from random selection to a greedier route selection the total airborne conflicts of the system reduce. At day 32, the experiment with inaccurate information has 61,998 conflicts / day, a daily reduction of 2,881 conflicts. At day 18, the experiment #2 has 63,601 conflicts / day, a reduction of 2,041 conflicts / day.

When the learning process becomes stable, there is a trend toward reduction of the total airborne conflicts, but the slope of the reduction is small compared to the reduction during the exploration period. The average airborne conflicts for the experiment using noisy information is 61,677 conflicts / day, whereas it is 62,259 conflicts / day for the experiment using accurate information. This difference of 0.9% with respect to mean of experiment using accurate local information indicates that performance remained the same when the
adaptable AOCs have either local or global real-time information.

In both experiments, there is variation of the total airborne conflicts between days even with a deterministic model of the NAS and without weather effects. As with fuel burn, these variations can only be attributed to the interaction of the learning processes of the AOCs. The standard deviation when AOCs have access to noisy local real-time information is 238, and it is 178 when the AOCs have accurate local information. This difference implies that inaccuracies tend to make predictability harder for the total airborne conflicts.

![Figure 4.27: Total airborne conflicts when the AOCs have accurate or noisy local real-time information.](image)

Figure 4.27: Total airborne conflicts when the AOCs have accurate or noisy local real-time information.

Figure 4.28 shows the evolution of the total departure delay for the experiments #6 and #2. The total departure delay starts at 127,630 minutes / day when the AOCs select routes randomly ($\varepsilon = 1$) and use noisy local real-time information. When the AOCs use accurate real-time local information, the total departure delay starts at 125,374 minutes /
day when the selections are done randomly. With the step-by-step change from random selection to a greedier selection the total departure delay of the system reduces, but shows significant variation in the process. At day 32, the simulation using noisy information has 120,459 minutes / day of departure delay, a reduction of 7,171 minutes / day. At day 18, the simulation using accurate information has 121,920 minutes / day, a reduction of 3,454 minutes / day.

When the learning process becomes stable, there is no indication of reduction in the total departure delay. The average departure delay for the experiment using noisy information is 118,250 minutes / day, whereas it is 118,604 minutes / day for the experiment using accurate information. This difference of 0.2% with respect to mean of experiment using accurate information indicates that there no effect on the total departure delay when the AOCs have access to noisy local real-time information.

The standard deviation when AOCs have access to noisy information is 3,151 minutes / day, and it is 2,927 minutes / day when the AOCs have accurate information. The standard deviations are two orders of magnitude smaller than the means, but the variance of the metric reduces when the AOCs have inaccurate global real-time information.

Figure 4.29 shows that the total arrival delay starts at about 572,535 minutes / day (8.45 minutes / flight) when the AOCs select routes randomly ($\varepsilon = 1$) with inaccurate local real-time information. When the information is accurate local, and real-time, the total arrival starts at about 570,463 minutes / day (8.42 minutes / flight). The incremental change from random selection to a greedier selection reduces the total arrival delay of the system. At day 32, the simulation using noisy information has 552,227 minutes / day of arrival delay (8.15 minutes / flight), a reduction of 20,308 minutes / day (17.98 second / flight). At day 18, the experiment using accurate information has 559,955 minutes / day (8.26 minutes / flight), a reduction of 10,508 minutes / day (9.31 seconds / flight).

When the learning process becomes stable, there is a trend toward reduction in the total arrival delay, but the slope of the trend is smaller than the slope in the exploration period. The average arrival delay for the experiment using noisy information is 549,369 minutes /
Figure 4.28: Total departure delay when the AOCs have accurate or noisy local real-time information. day (8.11 minutes / flight), whereas it is 556,644 minutes / day (8.21 minutes / flight) for the experiment using accurate information. This difference of 1.3% (6.4 second / flight) with respect to mean of experiment using accurate local information indicates that there is little effect on the total arrival delays when the AOCs have access to noisy local real-time information.

The standard deviation when AOCs have access to noisy local real-time information is 2,407 minutes / day, and it is 1,956 minutes / day when the AOCs have accurate information. The standard deviations are two orders of magnitude smaller than the means, but their values are different. The inaccuracies in the information tend to increment of the variance, which results in lower predictability.

Figure 4.30 shows the %OL at 74.1% when the AOCs select routes randomly ($\varepsilon = 1$) and have access to inaccurate local real-time information. When the AOCs have access
to accurate local real-time information and select routes randomly the %OL is 73.5% in the first simulated day. With the step-by-step change from random selection to a greedier selection the %OL of the system shows little reduction, but significant variation. At day 32, the simulation using inaccurate information is at 73.3% of %OL, a reduction of 0.8% or 11.1 simulated minutes less with at least one sector above its MAP value. At day 18, the experiment using accurate information (experiment #2) is at 72.5% of %OL, a reduction of 1.0% or 14.4 simulated minutes less with at least one sector above its MAP value. A difference of 14.4 minutes in 24 hours is an indication that the use of greedy route selection and adaptation has a small effect in the time distribution of sector congestion even in the case when the information is noisy, global, and real-time.

When the learning process becomes stable, there is no indication of reduction in the %OL. The average %OL for the experiment using noisy local real-time information is
72.38%, whereas it is 72.99% for the experiment using accurate information. This difference of 0.61% (8.8 simulated minutes) indicates that there is a reduction in the %OL when the AOCs have access to noisy local real-time information.

The standard deviations in the exploitation period are both less than 1%. The deviation is 0.409% when the AOCs have noisy information, and 0.385% when they have accurate information. This difference is about 30 seconds of simulated time in 24 simulated hours, which is an indication there is no change in variance when adaptable AOCs use noisy local real-time information for route selection.

Figure 4.30: %OL when the AOCs have accurate or noisy local real-time information.

Figure 4.31 suggests a lack of significant effect of the inaccuracies on the %Osch. The reason this apparent lack of effect of the independent variable is that congestion at the arrival airports is not changed by flightplan route selection, even when the information is noisy, global, and real-time.
Figure 4.31: %Osch when the AOCs have accurate or noisy local real-time information.

The comparison of experiments #6 and #2 uses the same steps used to compare experiments #5 and #1. If the datasets are input into the functions putting the dataset for experiment #6 first, and the dataset for experiment #2 second, all the parameters used for comparing experiments #5 and #1 are equal for this comparison. The null hypotheses in this comparison are also $H_0^5$ for the difference of mean values and $H_0^6$ for the difference of variances.

Table 4.17 shows that the standard deviations for the metrics are different between experiments for total airborne conflicts and departure delays. Nevertheless, no significant difference exists between the variances of the other metrics. In terms of predictability, the independent variable shows no effect in the metrics.

Table 4.18 shows that the presence of inaccuracies in the information used to select flightplan routes significantly reduces performance in terms of fuel burn, and %Osch. The other metrics do not show significant differences between experiments.
Table 4.17: Tests for equality of variances of the system-wide performance metrics when AOCs have local inaccurate information vs. local accurate information

<table>
<thead>
<tr>
<th>Metric</th>
<th>P-value&lt;sup&gt;a&lt;/sup&gt; (α = 0.05)</th>
<th>Ratio</th>
<th>95% CI</th>
<th>Statistic&lt;sup&gt;b&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>0.047</td>
<td>1.709</td>
<td>[1.007, 2.956]</td>
<td>1.709</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>0.031</td>
<td>1.789</td>
<td>[1.054, 3.094]</td>
<td>1.789</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>0.853</td>
<td>N/A</td>
<td>N/A</td>
<td>0.034</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>0.511</td>
<td>N/A</td>
<td>N/A</td>
<td>0.435</td>
</tr>
<tr>
<td>%OL</td>
<td>0.652</td>
<td>1.127</td>
<td>[0.664, 1.950]</td>
<td>1.127</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.763</td>
<td>N/A</td>
<td>N/A</td>
<td>0.091</td>
</tr>
</tbody>
</table>

<sup>a</sup> Null hypothesis rejected if p < α.  <sup>b</sup> Using F-test or Levene’s test for the comparison of variances.

Table 4.18: Hypothesis testing for the means of the system-wide metrics when AOCs have local inaccurate information vs. local accurate information

<table>
<thead>
<tr>
<th>Metric</th>
<th>P-value&lt;sup&gt;a&lt;/sup&gt; (α = 0.05)</th>
<th>Difference (%)</th>
<th>95% CI</th>
<th>Statistic&lt;sup&gt;b&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>0.000</td>
<td>506,895 (0.2%)</td>
<td>[482,236, ∞]</td>
<td>t = 34.200</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>1.000</td>
<td>-583 (-0.9%)</td>
<td>[-650, ∞]</td>
<td>t = -14.300</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>0.754</td>
<td>-260 (-0.2%)</td>
<td>[-921, ∞]</td>
<td>W = 1,427</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>1.000</td>
<td>-7,275 (-1.3%)</td>
<td>[-7,958, ∞]</td>
<td>t = -17.600</td>
</tr>
<tr>
<td>%OL</td>
<td>1.000</td>
<td>-0.61 (-0.6%)</td>
<td>[-0.73, ∞]</td>
<td>t = -8.053</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.000</td>
<td>0.02 (0.0%)</td>
<td>[0.01, ∞]</td>
<td>W = 2,761</td>
</tr>
</tbody>
</table>

<sup>a</sup> Null hypothesis rejected if p < α.  <sup>b</sup> Using unpaired t-test or Wilcoxon’s test for the difference between means.
Table 4.19: Hypothesis testing for the variance of the system-wide performance metrics when AOCs have local inaccurate information vs. local accurate information

<table>
<thead>
<tr>
<th>Metric</th>
<th>P-value $^a$ $(\alpha = 0.05)$</th>
<th>Ratio (%)</th>
<th>95% CI</th>
<th>Statistic $^b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>0.024</td>
<td>1.709 (71%)</td>
<td>[1.097, $\infty$]</td>
<td>1.709</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>0.016</td>
<td>1.789 (79%)</td>
<td>[1.148, $\infty$]</td>
<td>1.789</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>0.290</td>
<td>1.159 (16%)</td>
<td>[0.744, $\infty$]</td>
<td>1.159</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>0.062</td>
<td>1.513 (51%)</td>
<td>[0.971, $\infty$]</td>
<td>1.513</td>
</tr>
<tr>
<td>%OL</td>
<td>0.326</td>
<td>1.127 (13%)</td>
<td>[0.723, $\infty$]</td>
<td>1.127</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.490</td>
<td>1.004 (0%)</td>
<td>[0.644, $\infty$]</td>
<td>1.004</td>
</tr>
</tbody>
</table>

$^a$ Null hypothesis rejected if $p < \alpha$. $^b$ Using F-test for the ratio of variances.

Table 4.19 shows that the null hypothesis is rejected, with 95% confidence, only for the total airborne conflicts. For all the other metrics there is no evidence to say that the variances are significantly different.
Chapter 5: Conclusions

This section summarizes the results of the case study, discusses the analysis of the results, explains the limitations of this study, and describes future work.

5.1 Summary of Results

This section summarizes the results of the case study for the time to reach steady-state, the improvements during the exploration period, and the effect of the independent variables on the performance (i.e., mean of the performance metrics) and variance during the steady-state.

5.1.1 Steady-State

The first column of Table 5.1 shows that knowledge is acquired faster when the information is real-time, than when it is delayed. This delay in achieving steady-state is a consequence of the use of historic records in the route selection process. The availability of global information and the accuracy of the data do not affect the speed of knowledge acquisition.

Table 5.1: Number of simulated days needed by the experiments to reach the steady-state

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Reaches 60% of non-zero q-records (Simulated days)</th>
<th>Reaches 10% of q-records changing 10% or less (Simulated days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global data</td>
<td>14</td>
<td>17</td>
</tr>
<tr>
<td>Local data</td>
<td>14</td>
<td>18</td>
</tr>
<tr>
<td>Delayed global data</td>
<td>17</td>
<td>21</td>
</tr>
<tr>
<td>Delayed local data</td>
<td>17</td>
<td>26</td>
</tr>
<tr>
<td>Noisy global data</td>
<td>14</td>
<td>23</td>
</tr>
<tr>
<td>Noisy local data</td>
<td>14</td>
<td>32</td>
</tr>
</tbody>
</table>
The second column of Table 5.1 shows that the learning process reaches steady-state faster when the information is global, accurate, and real-time (first row) than in any other combination of values for the independent variables. The absence of global information, the presence of latency in the data, and the presence of inaccuracies increase the time to reach steady-state. The last row of the table shows that the learning process takes the longest time when the information is local, real-time, and noisy.

5.1.2 The Exploration Period

Table 5.2 summarizes the reductions in the system-wide metrics between the first day of simulation (i.e., pure random flightplan route selection) and the start of the steady-state for all the experiments and metrics. The total fuel burn, total airborne conflicts, total departure delay, total arrival delay, and %OL exhibit reductions that are due to the inclusion of adaptation in the decision-making process of the AOCs. %OSch does not change during the exploration period.

5.1.3 Effect of the Independent Variables on System Performance

Table 5.3 summarizes the effect of the independent variables on system performance. The availability of global information reduces only total airborne conflicts of the system. It does not have a statistically significant effect in any of the other metrics.

The presence of latency marginally increases the conflicts and the delays of the system. Exceptions are the total fuel burn and %Osch which are not significantly affected by the availability of global information.

The presence of inaccuracies in the data (+/-30% random errors) increases the fuel burn. The total airborne conflicts, and the arrival delays show statistically significant reductions with global and with local information. The total departure delay, %OL, and the %Osch the inaccuracies decrease but, the magnitudes of the changes are not statistically significant.
Table 5.2: Reductions in the system-wide performance metrics between the pure random flightplan route selection and the start of the steady-state

<table>
<thead>
<tr>
<th>Metric</th>
<th>Experiment</th>
<th>Absolute reduction</th>
<th>% reduction</th>
<th>Detailed reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total fuel burn</td>
<td>1</td>
<td>471,353 kg/day</td>
<td>0.20</td>
<td>9.2 kg/flight/day</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>664,084 kg/day</td>
<td>0.20</td>
<td>13.0 kg/flight/day</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>442,374 kg/day</td>
<td>0.10</td>
<td>8.7 kg/flight/day</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>752,935 kg/day</td>
<td>0.30</td>
<td>14.8 kg/flight/day</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>140,116 kg/day</td>
<td>0.00</td>
<td>2.7 kg/flight/day</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>-85,097 kg/day</td>
<td>0.00</td>
<td>-1.7 kg/flight/day</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>1</td>
<td>4.482</td>
<td>6.90</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.041</td>
<td>3.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3.608</td>
<td>5.60</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2.586</td>
<td>4.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5.160</td>
<td>8.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2.881</td>
<td>4.40</td>
<td></td>
</tr>
<tr>
<td>Total departure delay</td>
<td>1</td>
<td>5,463 min</td>
<td>4.30</td>
<td>4.8 secs/flight</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3,454 min</td>
<td>2.80</td>
<td>3.0 secs/flight</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4,130 min</td>
<td>3.30</td>
<td>3.7 secs/flight</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3,436 min</td>
<td>2.80</td>
<td>3.0 secs/flight</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>3,391 min</td>
<td>2.70</td>
<td>3.0 secs/flight</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>7,171 min</td>
<td>5.60</td>
<td>6.4 secs/flight</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>1</td>
<td>11,420 min</td>
<td>2.00</td>
<td>10.1 secs/flight</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>10,508 min</td>
<td>1.80</td>
<td>9.3 secs/flight</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>9,512 min</td>
<td>1.70</td>
<td>8.4 secs/flight</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>10,673 min</td>
<td>1.90</td>
<td>9.5 secs/flight</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>13,106 min</td>
<td>2.30</td>
<td>11.6 secs/flight</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>20,308 min</td>
<td>3.50</td>
<td>18.0 secs/flight</td>
</tr>
<tr>
<td>%OL</td>
<td>1</td>
<td>1.1</td>
<td></td>
<td>16.0 min/day</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.2</td>
<td></td>
<td>17.0 min/day</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>1.5</td>
<td></td>
<td>21.0 min/day</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.5</td>
<td></td>
<td>7.1 min/day</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>3.1</td>
<td></td>
<td>44.1 min/day</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2.5</td>
<td></td>
<td>36.0 min/day</td>
</tr>
<tr>
<td>% Osch</td>
<td>1</td>
<td>0.0 %</td>
<td>0.0</td>
<td>0 min/day</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.0</td>
<td>0.0</td>
<td>0 min/day</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.0</td>
<td>0.0</td>
<td>0 min/day</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.0</td>
<td>0.0</td>
<td>0 min/day</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.0</td>
<td>0.0</td>
<td>0 min/day</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>0.0</td>
<td>0.0</td>
<td>0 min/day</td>
</tr>
</tbody>
</table>
Table 5.3: Effect of the independent variable on the mean of the system-wide performance metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Difference of mean (%)</th>
<th>Availability</th>
<th>Latency</th>
<th>Inaccuracies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Global data</td>
<td>Global data</td>
<td>Local data</td>
</tr>
<tr>
<td>Total fuel burn</td>
<td>174,607 (0.10%)</td>
<td>18,099 (0.00%)</td>
<td>-9,553 (-0.00%)</td>
<td>401,362 (0.11%)</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>-1,713 (-2.80%)</td>
<td>127 (0.18%)</td>
<td>409 (0.70%)</td>
<td>-2,247 (-3.70%)</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>81 (0.10%)</td>
<td>2,384 (2.01%)</td>
<td>462 (0.54%)</td>
<td>-44 (-0.02%)</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>1,341 (0.20%)</td>
<td>1,698 (0.28%)</td>
<td>711 (0.10%)</td>
<td>-5,857 (-1.00%)</td>
</tr>
<tr>
<td>%OL</td>
<td>0.40 (0.40%)</td>
<td>-0.19 (-0.19%)</td>
<td>0.37 (0.37%)</td>
<td>-1.21 (-1.21%)</td>
</tr>
<tr>
<td>%Osch</td>
<td>0.00 (0.00%)</td>
<td>0.01 (0.01%)</td>
<td>0.01 (0.01%)</td>
<td>0.00 (0.00%)</td>
</tr>
</tbody>
</table>

* Statistically significant increase  b Statistically significant reduction

5.1.4 Effect of the Independent Variables on the Variance of the System Performance

Table 5.4 summarizes the effect of the independent variables on the variances of the metrics in the steady-state period (i.e., exploitation) of the simulation. The presence of global information (column labeled “Availability”) decreases variance of the arrival delays, but increases the variance of the airborne conflicts. The variance of the other metrics is statistically equal when global and local data are available.

The presence of latency reduces the variances when the AOCs have global delayed information. When the AOCs have local delayed information, the variances of fuel burn, and conflicts increase. However, only the total departure delay shows significant differences of 66% (reduction) with global information, and 83% (reduction) with local information. The arrival delay shows a significant reduction of 59% when local information is available.

The presence of inaccuracies in the data has more effect in the variances of airborne conflicts (with global and local data), departure delay, arrival delay, and %OL (only with global data). The effect is stronger when global information is available.

5.2 Discussion of Analysis

This section starts with a discussion of the behavior in the *exploration period* of the simulations, in which the AOCs starts selecting routes *randomly* and progressively move toward a
Table 5.4: Effect of the independent variables on the variance of the system-wide performance metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>% difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Availability</td>
</tr>
<tr>
<td></td>
<td>Global data</td>
</tr>
<tr>
<td>Total fuel burn</td>
<td>50.3</td>
</tr>
<tr>
<td>Total airborne conflicts</td>
<td>264.2</td>
</tr>
<tr>
<td>Total departure delay</td>
<td>-27.9</td>
</tr>
<tr>
<td>Total arrival delay</td>
<td>-37.6</td>
</tr>
<tr>
<td>%OL</td>
<td>45.25</td>
</tr>
<tr>
<td>%Osch</td>
<td>4.10</td>
</tr>
</tbody>
</table>

*a Statistically significant increase  b Statistically significant reduction

greeder selection strategy supported by the knowledge AOCs acquire in the process. Next, the discussion focuses on explaining the observations for the exploitation period that starts when the learning process of the AOCs reaches steady-state. The discussion explains the observations of all the independent variables separately. Finally, the advantages of including adaptation in the flightplan route selection are presented, and implications to the SWIM (and NextGen) are described.

5.2.1 The Exploration Period

The exploration period of the simulations starts at the first simulated day, when the AOCs have no knowledge and select flightplan routes randomly from sets of alternatives, and ends when the learning process of the AOC reaches steady-state.

Table 5.2 on page 159 indicates that total fuel burn, total airborne conflicts, total departure delay, total arrival delay, and %OL reduce their values during the exploration period while the AOCs gradually change from purely random route selection (i.e., on the first simulated day) to mostly greedy route selection (i.e., after the steady-state is reached). The improvements are evidence of the benefit of using adaptation for the flightplan route selection. All AOCs act independently to determine the best routes and the result is an improvement of the system regardless of moderate errors in SWIM (e.g., lack or retention
of information, corruptions or concealment of information, and delay of the data).

The metric for arrival airport congestion (%Osch) does not improve during the exploration period, which confirms previous evidence that congestion at the destination airports cannot be mitigated by flightplan route selection [Calderon-Meza and Sherry, 2010a]. The marginal reductions in the sector congestion metric (%OL) suggest that the metric is not adequate for this problem since it is a first order approximation to the temporal distribution of sector congestion, but it does not consider the spatial distribution of the congestion.

The system is in exploitation period after the learning process of the AOCs reaches stability. When the learning process is stable the AOCs select flightplan routes greedily (i.e., the best route they know at the moment) with sporadic random route selections. In the exploitation period, the system performance metrics enter into a “steady-state”. Some of the charts for the metrics suggest time dependencies between during the steady-state. Table 5.5 shows the computation of linear regressions for all the metrics of all experiments during the steady-state. As indicated in the table, 23 of the 36 regressions are statistically significant (i.e., the regression is a statistically valid model for the data). The $R^2$ value describes how well the model fits the data. Only fuel burn, and conflicts have $R^2 > 40\%$ for all the experiments. Arrival delay shows values smaller than 20\% for the first two experiments, and values greater than 40\% for the other experiments. The other metrics show low values for $R^2$ despite of the validity of the model. For the 13 regressions that are not statically significant (i.e., the regressions do not model the data) the data points were considered IID for the analysis since they do not show any evidence of time dependence.

Despite the time dependencies implied by the high values of $p_x$ and $R^2$, all the slopes of the statistically valid regression models are several orders of magnitude smaller than the corresponding slopes (see the third and fourth columns of Table 5.5), which indicates that the improvements$^1$ observed in the metrics during the steady-state are marginal. Due to the marginality of improvements the data points (i.e., metric values) in the steady-state

$^1$The word improvement is used because all slopes of the valid models are negative, and reductions in the metrics are performance improvements in this simulations.
were considered IID for the purpose of the analysis.

5.2.2 Effects of Global Information

The comparison of performance between the first simulated day and the exploitation period indicates that the use of adaptation results in system-wide performance improvements with respect to random flightplan route selection.

The significant reduction in the total number of airborne conflicts when global information is available occurs because the computation of rewards used in the learning process uses conflicts data only when global information is available. With local information the number of conflicts for the flights is unknown to the AOCs. The marginal differences between experiments in the other metrics imply that the alternate routes available in the repository are different enough to reduce the conflicts, but insufficient to impact the delays, congestion, or fuel burn.

The significant increase in the variance of the airborne conflicts observed when global information is available can be explained by the increase in dimensionality of the search space. With local information, the search space has four dimensions because conflicts and congestion in sectors are not considered. With global information, however, the search space has six dimensions, and it also contains data from other airlines, which makes more difficult the exploration of the search space. The statistical equality of the variances of the other metrics implies that the use of adaptation for flightplan route selection creates a negative feedback loop that maintains the system in steady-state when more information is available to the airlines.

5.2.3 Effects of Latency in the Information

The main effect of latency in the information is that the learning process of the AOCs requires a longer time to stabilize. Additionally, the presence of moderate latency in the communication of information used for flightplan route selection decreases the performance of the NAS. The decrease in historical data available to the AOCs caused by the latency
Table 5.5: Effect of the independent variables on the variance of the system-wide performance metrics

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Regression equation&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Y Intercept (b)</th>
<th>Slope (m)</th>
<th>R²</th>
<th>p&lt;sub&gt;x&lt;/sub&gt;&lt;sup&gt;b&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fuel burn</td>
<td>299,706,349</td>
<td>-2,872</td>
<td>43.9%</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>Conflicts</td>
<td>61,302</td>
<td>-15.6</td>
<td>72.7%</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>Departure delay</td>
<td>118,526</td>
<td>+8.6</td>
<td>0.4%</td>
<td>0.612</td>
</tr>
<tr>
<td></td>
<td>Arrival delay</td>
<td>559,575</td>
<td>-32.7</td>
<td>15.5%</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>%OL</td>
<td>73.65</td>
<td>-0.0054</td>
<td>4.7%</td>
<td>0.084</td>
</tr>
<tr>
<td></td>
<td>%Osch</td>
<td>10.60</td>
<td>-0.0001</td>
<td>3.0%</td>
<td>0.173</td>
</tr>
<tr>
<td>Global, real-time, accurate (#1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global, delayed, accurate (#3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global, real-time, noisy (#5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Local, real-time, accurate (#2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Local, delayed, accurate (#4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Local, real-time, noisy (#6)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<sup>a</sup> The regression equation is: \( Y = m \times \text{day} + b \).  
<sup>b</sup> Regression is a statistically valid model if \( p_x < 0.05 \).
hinders the decision process and is the cause of the statistically significant decrease in system-wide performance. The decrease in performance is greater when the AOCs have local information available than when they have global information. The use of global information causes the AOCs learning process to be more resilient to latency. From a Control Systems perspective, using global information has the same effect as improving the feedback loop of the system, or adding one more feedback loop to the system.

The variance of the system performance metrics does not increase with the presence of latency. The learning process adapts successfully to the delays in the information. Since the learning process requires several repetitions of a situation to “learn” the correct action to take in that situation, a delay of 1 day does not force the AOCs to test more alternate routes, the same routes are tested and the variation remains statistically equal in all the cases.

5.2.4 Effects of Inaccuracies in the Information

The reason for the apparent lack of effect of inaccuracies in information on the performance of the system can be that the feedback loop created by the learning process works as a filter for the noise that causes the inaccuracies. Since the RL process is designed to learn the trends instead of the “noise” of a system, there is little impact of high frequency signals. Conflicts and arrival delays increased their variances, especially when the information was global. The other four metrics showed no significant changes or reduced their variance regardless of the presence or absence of noise: the RL process has the same compensating effect on the variance that it had on the performance. An alternate experiment using +/-10% inaccuracies did not show significant effects in system-wide performance, which supports the idea that noise is filtered by the RL process.

The adaptation used in the flightplan route selection improves robustness, by compensating negative effects of latency and inaccuracies (see section 2.3.3 on page 24). The inclusion of global information acts as an improvement of the feedback loop by further reducing the negative effects of latency and inaccuracy instead of generating information overload.
However, the global information can cause an increase in the variances because the system must optimize for additional criteria and is forced to experiment with more alternatives to learn the optimal values. An implication for SWIM is that the system does not become over-dependent on the availability, exactitude, and timeliness information infrastructure if the AOCs are able to adapt.

5.2.5 Extensions to the Analysis

The analysis of the results focused on the system-wide effects of SWIM in the presence of adaptive flightplan route selection. Future work can extend the analysis to provide more details, e.g. analyze individual airlines, routes, or regions the NAS.

One approach to extending the analysis is to determine the contribution of each flight performance metric (see Section 3.4.3 on page 82) to the system performance, and the way AOCs adapt to the changing circumstances during the day of operations. A procedure to determine the contributions is described as follows:

1. Select an O/D pair that has several flights a day (e.g. an O/D pair serviced by two or more airlines)

2. Group historic values of the metrics for flights for the O/D pair by departure times considering close departure times as equal

3. For each of the flight performance metrics of distance, number of conflicts, departure delay, arrival delay, and percentage of overloaded sectors crossed during the flight, compute the average for each departure time

4. Compute the daily ranges of the averages for each metric

5. Normalize the average for each metric and departure time in such a way that a normalized value of 0 means that the average equals the minimum of the range, and a value of 1 means that it equals the maximum
6. Add the normalized values of the metrics for each departure time to obtain a total benefits value that can range from 0, if the all the averages equal their minima, to 5, if all averages equal their maxima.

7. Compute the percentage contribution of each metric to the total benefits by dividing the each normalized average by the total benefits.

Figure 5.1: Contribution of each metric to the total benefits during the day for the LAX-SEA O/D pair.

Figure 5.1 is an example for the Los Angeles International (LAX)- Seattle/Tacoma International (SEA) O/D pair that has 13 flights a day. The data used to obtain the figure correspond to the experiment with global, real-time, and accurate information (experiment #1). The figure shows that airlines do obtain different percentage of benefits for each metric for the same route during the day. The airborne conflicts are minimized at 9:30 UTC (i.e. 1:30 am PST), and they obtain their worst value at 21:17 UTC (i.e. 1:30 pm PST). The departure delay obtains the best results after 23:47 UTC, and 4:13 UTC.

Table 5.6 contains the historically dominant routes for each time of the day for the LAX/SEA O/D pair. The data used to obtain this table includes 80 simulated days for the experiment with global, real-time, and accurate information (i.e. the table corresponds to the Figure 5.1). The table shows that there is always a dominant route for each time of
Table 5.6: Dominant routes for each time of the day when AOCs have global, real-time, and accurate information

<table>
<thead>
<tr>
<th>Time (hh:mm UTC)</th>
<th>Route identifier</th>
<th>Route string</th>
<th>Distance (nm)</th>
<th>Selection percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:55</td>
<td>42065</td>
<td>LAX./.3554N/11918W..BTG..OLM5.SEA</td>
<td>848</td>
<td>78%</td>
</tr>
<tr>
<td>2:12</td>
<td>42065</td>
<td>LAX./.3554N/11918W..BTG..OLM5.SEA</td>
<td>848</td>
<td>73%</td>
</tr>
<tr>
<td>3:12</td>
<td>49074</td>
<td>LAX./.3555N/11918W..BTG..OLM5.SEA</td>
<td>845</td>
<td>81%</td>
</tr>
<tr>
<td>3:59</td>
<td>45489</td>
<td>LAX./.ECA075020..BTG..OLM..OLM5..SEA</td>
<td>854</td>
<td>70%</td>
</tr>
<tr>
<td>4:13</td>
<td>43319</td>
<td>LAX./.GRAPE..BTG..OLM..OLM5..SEA</td>
<td>848</td>
<td>74%</td>
</tr>
<tr>
<td>5:21</td>
<td>49076</td>
<td>LAX./.DUCKE..LMT..BTG..OLM5..SEA</td>
<td>849</td>
<td>75%</td>
</tr>
<tr>
<td>5:48</td>
<td>45491</td>
<td>LAX./.GM2..BTG..OLM5..SEA</td>
<td>841</td>
<td>78%</td>
</tr>
<tr>
<td>9:30</td>
<td>38137</td>
<td>LAX.VTU5.RZS..AVE..FMG..J5..SEA..SEA</td>
<td>869</td>
<td>68%</td>
</tr>
<tr>
<td>13:26</td>
<td>102040</td>
<td>LAX..SEA</td>
<td>830</td>
<td>76%</td>
</tr>
<tr>
<td>13:44</td>
<td>40315</td>
<td>LAX.GMN4.EHF..J65..CZQ..LIN..JT89..BTG..OLM5..SEA</td>
<td>857</td>
<td>75%</td>
</tr>
<tr>
<td>14:07</td>
<td>42065</td>
<td>LAX./.3554N/11918W..BTG..OLM5..SEA</td>
<td>848</td>
<td>70%</td>
</tr>
<tr>
<td>17:25</td>
<td>42065</td>
<td>LAX./.3554N/11918W..BTG..OLM5..SEA</td>
<td>848</td>
<td>79%</td>
</tr>
<tr>
<td>18:35</td>
<td>43319</td>
<td>LAX./.GRAPE..BTG..OLM5..SEA</td>
<td>848</td>
<td>70%</td>
</tr>
<tr>
<td>21:12</td>
<td>40312</td>
<td>LAX./.EHF184002..BTG..OLM5..SEA</td>
<td>848</td>
<td>75%</td>
</tr>
<tr>
<td>21:27</td>
<td>40313</td>
<td>LAX./.GMN..DUCKE..LMT..BTG..OLM5..SEA</td>
<td>846</td>
<td>70%</td>
</tr>
<tr>
<td>23:17</td>
<td>45487</td>
<td>LAX./.DUCKE..BORDY..BTG..OLM5..SEA</td>
<td>848</td>
<td>74%</td>
</tr>
<tr>
<td>23:47</td>
<td>45494</td>
<td>LAX.VTU5.RZS..J88..SNS..RBL..BTG..OLM5..SEA</td>
<td>891</td>
<td>65%</td>
</tr>
<tr>
<td>23:50</td>
<td>40313</td>
<td>LAX./.GMN..DUCKE..LMT..BTG..OLM5..SEA</td>
<td>846</td>
<td>79%</td>
</tr>
</tbody>
</table>

The day, but the route is not the same throughout the day. The shortest route (i.e. GCD route) is dominant only at 13:26, the same time in which Figure 5.1 indicates that distance is obtaining the most benefits.

Figure 5.2 shows that the contributions of the metrics for the experiment with local, real-time, and accurate information are not the same as in the Figure 5.1. The airborne conflicts are at their best value at 18:35 UTC, and at the same time, departure and arrival delays are at their worst values. The best value in distance is at 17:25 UTC, the same time in which the percentage of congested sectors crossed is at its worst value.

This extension of the analysis (for all experiments and more O/D pairs) provides more insight about the trade-offs between performance metrics and how the adaptable route selection process of the AOCs handle the trade-offs.
Figure 5.2: Contribution of each metrics to the total benefits during the day for the LAX-SEA O/D pair when AOCs have local, real-time, accurate data.

5.2.6 Summary of contributions

The contributions of this dissertation can be summarized as follows:

- Design and development of the architecture and algorithms to introduce airline adaptive decision-making into NAS-wide simulators (such as FACET and Airspace Concept Evaluation System (ACES)).

- Successful integration of a NAS-wide simulator, FACET, an ABS tool, MASON, and the airline adaptive decision-making.

- Development of a methodology for analysis of the results of adaptive behaviors in the NAS.

- Analysis of robustness of adaptive flightplan route selection to degradation of SWIM functionality.

5.3 Limitations and Future Work

This section discusses limitations of the input data, performance metrics, stochasticity of the simulations, the set of alternate routes, and the execution time of the simulations. In
addition, the economical significance of the results, future applications of the approach, and the validation of the approach are discussed.

5.3.1 Performance Metrics

The metrics %OL and %Osch show little change in all experiments regardless of the values of the independent variables and the period of the simulation (i.e., exploration or exploitation). This apparent lack of effect of the independent variables indicates that the metrics do not contribute with information to the adaptation process of the AOCs. Future work should experiment with metrics for airspace complexity, congestion, and temporal/spatial distribution of the traffic.

5.3.2 Improvements in the Computation of Rewards

The comparison of performance metrics in the computation of rewards assume absolute precision of the outcomes from FACET (see Equation 3.3 on page 91). To mitigate the effects of the absolute precision assumption in the performance metrics, the comparison could be modified. One possible modification could be the introduction of thresholds in the comparisons. For instance, instead of comparing fuel as $f_t \geq f_{t-1}$, the comparison could be $|f_t - f_{t-1}| < \text{threshold}$. One issue with this approach is the need to calibrate to reflect economical relevance of the metrics for each airline. The thresholds may be different between airlines, could change through time, and must be elicited or inferred from other types of analyses.

Another possibility for the mitigation of the absolute precision assumption is to round the values of the performance metrics so that the least significant digits have no effect in the comparisons. As with the threshold, rounding should be calibrated to keep it from filtering out relevant values.
5.3.3 Improvements to the Input File

The simulation does not have access to the schedule of the flights, which implies that the scheduled departure time is equal to the wheels-off time, and the scheduled arrival time can only be estimated in the simulation (see Algorithm 3.4 on page 84). Therefore, the departure and arrival delays are also estimations. Furthermore, the current input file, since it contains tracking data, is the result of the work of the ATC during the operations: departures and arrivals are synchronized by the controllers and GDPs, and traffic through sectors is balanced. The effect of this implicit synchronization and balance results lower delays and less congestion in sectors and at airports. Future work could generate a new input file that is closer to the schedule.

5.3.4 Compression of the Q-Matrix

The total number of Q-records in the simulation is about 320,000. A considerable amount of memory is needed to store the records during the simulation, and execution time is spent reading the Q-records from the database, and updating the Q-values at the end of the simulation. The use of compression techniques for the Q-matrix can reduce the memory and time while maintaining the fidelity of the information [Balakrishna, 2009].

5.3.5 Stochasticity in NAS-Wide Simulation

The experiments in this dissertation did not include weather effects or other sources of stochasticity like push-back delays. As a result, FACET simulations model a deterministic NAS. The assumption of a deterministic NAS is convenient to isolate the effect of the adaptation on the route selection process and to simplify its evaluation. However, future work could introduce stochasticity in the NAS and evaluate adaptation in a more realistic scenario. The inclusion of weather can be approached in several ways. First, FACET can receive input for winds that would add one aspect of weather to the simulation. Second, the effects of weather could be modeled by reductions in capacities for sectors and airports. Third, the code of the Main Application already includes the functionality for exponentially
distributed push-back delays (see Figure 3.10 on page 56), which can be activated using a parameter in the configuration file (see section 3.4.2 on page 73).

5.3.6 Alternate Routes

The current simulation obtains alternate routes from a database. The Main Application (see section 3.2 on page 48) has functionality to expand the database when the input file contains new routes, the number of alternate routes remains constant when the same input file is repeatedly used as is the case in the simulations in this dissertation.

Figure 5.3: Histogram for difference of route distance with respect to corresponding GCD route.

Figure 5.3 shows that most of the alternate routes in the database are close to the GCD for the origin / destination pairs. The average distance difference with respect to the GCD is 28.0 miles for the current database. In future work, when the capacity of sectors is reduced to model weather effects, an alternate route synthesis algorithm could be required[Slattery
and Zhao, 1995]. In those cases, alternate routes that avoid closed or very low capacity sectors could be the only option.

5.3.7 Economical Significance

Even though some comparisons between experiments show statistically significant differences, the analysis in this dissertation did not consider the economical significance or relevance of the results. Future work could explore options to relate the performance metrics to economic metrics and determine their relevance for the stakeholders.

5.3.8 Collaborative Trajectory Options Program

The FAA is working to introduce the CTOP in the NAS. The similarity between the adaptive flightplan route selection presented in this dissertation and a CTOP opens an opportunity for applying this research to a practical problem. Future work can refine the adaptive flightplan route selection process so that the AOCs can use it to compute the TOSs required by the FAA for a CTOP.

The future refinements must consider that a CTOP is a regional, limited in time, deterministic, centralized, and collaborative decision-making process. In contrast, the flightplan route selection process described in this dissertation is NAS-wide, adaptable, distributed, and shows only implicit collaboration through the information sharing provided by SWIM.

5.3.9 Simulation Execution Time

The execution of each simulation (i.e., 1 day of operations) takes approximately 2 hours and 30 minutes running on a 4*Xeon 64bit @ 2.66GHz, 16 GB RAM computer. To apply this approach to real-world situations, the execution time must be reduced considerably. For instance, in the case of a CTOP, AOC’s would need to submit their TOS’s to the FAA several hours after the FCA is defined (i.e. initial step of the CTOP). The current simulator requires several simulations to enter the stable state and provide valid preference values for alternate routes. Future work could explore different software architectures and algorithms,
as well as hardware, to shorten the execution time of the simulations. One possible first step in that direction is to separate the interaction with the dataset from the thread of execution of the Main Application. The creation of messages queues in separate computers that create bridges between the simulation and the database is foreseeable as a performance improvement. These message queues should be accompanied by keeping track of as many values as possible in memory so that the frequency and volume of the interactions with the message queue (i.e. database) is reduced.

5.3.10 Validation of the Approach and the Results

An approach to solve or analyze a problem does not receive the name “methodology” until it has been demonstrated in several domains with sufficient examples, and it has demonstrated its effectiveness. Due to the complexity of the NAS and the interactions of its agents, and time needed to apply the approach to other domains compared to the time available for a dissertation process, the approach in this dissertation has not been demonstrated in other domains. This validation process is left as future work. Furthermore, the validation of the results is a challenging task too. Some of the data used to select flightplan routes are not available to the airlines in the current NAS. Therefore, the validation must be internal first. In future work, the simulation could be implemented using other NAS-wide simulators or different programming languages. If the results obtained with the other simulations are similar to the results presented in this dissertation, there would be evidence of the validity of the results from this research.
Appendix A: Configuration Files and Scripts for Execution and Analysis

This section contains the configuration files and scripts used to create the database, extract data from the database, create charts, test for normality, and to test for the equality of the means and variances.

A.1 SQL Scripts

The following script creates the “dissertation” database used to execute a simulation. To execute the script open a terminal in the Linux computer that has MySQL server installed and running. On the terminal type “mysql”. A prompt ("mysql>") should appear indicating the successful connection to the database server. Type each one of the following lines on the prompt:

```
CREATE DATABASE 'dissertation' /*!40100 DEFAULT CHARACTER SET latin1 */;
CREATE USER 'dissertation'@'%' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON 'dissertation' TO 'dissertation'@'%;'
```

The script creates a user called “dissertation” that has all the privileges on the database “dissertation”, and uses a password (that must be typed instead of the word ’password’). This user and password must be used in the configuration files for the experiments.

The following script creates all the tables needed in the database dissertation to execute a simulation.

```
-- MySQL dump 10.13  Distrib 5.1.48, for redhat-linux-gnu (x86_64)
-- Host: localhost  Database: dissertation

-- Server version 5.1.48

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
```
CREATE DATABASE IF NOT EXISTS dissertation;

USE DATABASE dissertation;

-- Table structure for table 'airline'

DROP TABLE IF EXISTS 'airline';
/*140101 SET @saved_cs_client = @@character_set_client */;
/*140101 SET character_set_client = utf8 */;
CREATE TABLE 'airline' (  
'ICAACODE' varchar(3) NOT NULL,
'ICAO_CODE' varchar(3) DEFAULT NULL,
'NAME' varchar(100) DEFAULT NULL,
'COUNTRY_CODE' smallint(4) DEFAULT NULL,
'BTS_AIRLINE_ID' smallint(6) DEFAULT NULL,
'START_DATE' datetime DEFAULT NULL,
'END_DATE' datetime DEFAULT NULL,
PRIMARY KEY ('ICAACODE'),
KEY 'ICAO_CODE_IDX' ('ICAO_CODE'),
KEY 'COUNTRY_CODE_IDX' ('COUNTRY_CODE')
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Airlines, code, names, and dates';
/*140101 SET character_set_client = @saved_cs_client */;

-- Table structure for table 'routes'

DROP TABLE IF EXISTS 'routes';
/*140101 SET @saved_cs_client = @@character_set_client */;
/*140101 SET character_set_client = utf8 */;
CREATE TABLE 'routes' (  
'seq' mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
'origin' varchar(15) NOT NULL,
'dest' varchar(15) NOT NULL,
'fp' varchar(600) NOT NULL COMMENT 'Flight plans (most_be_unique)',
'distance' double DEFAULT NULL,
PRIMARY KEY ('seq'),
KEY 'origin' ('origin'),
KEY 'dest' ('dest')
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=latin1;
/*140101 SET character_set_client = @saved_cs_client */;
-- Table structure for table 'executions_sequence'

DROP TABLE IF EXISTS 'executions_sequence';
//@40101 SET @saved_cs_client = @@character_set_client ;
//@40101 SET character_set_client = utf8 ;
CREATE TABLE 'executions_sequence' ( 
  'TREATMENT' smallint(6) NOT NULL,
  'EXECUTION' smallint(6) NOT NULL,
  'FINISHED' tinyint(1) unsigned NOT NULL DEFAULT '0',
  'TOT_FLIGHTS' mediumint(8) unsigned NOT NULL DEFAULT '0',
  'DECISION_FLIGHTS' mediumint(8) unsigned NOT NULL DEFAULT '0',
  'TOT_AIRLINES' smallint(4) unsigned NOT NULL DEFAULT '0',
  'DURATION' float unsigned NOT NULL DEFAULT '0',
  'TOT_ROUTES' mediumint(8) unsigned NOT NULL DEFAULT '0',
  'QRECS_NON_ZERO' mediumint(8) unsigned NOT NULL DEFAULT '0',
  'EPSILON' float unsigned NOT NULL DEFAULT '0',
  'LAMBDA' float unsigned NOT NULL DEFAULT '0',
  PRIMARY KEY ('TREATMENT', 'EXECUTION') ) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='Counts the execution (unique number)';
//@40101 SET character_set_client = @saved_cs_client ;

-- Table structure for table 'airline_benefit'

DROP TABLE IF EXISTS 'airline_benefit';
//@40101 SET @saved_cs_client = @@character_set_client ;
//@40101 SET character_set_client = utf8 ;
CREATE TABLE 'airline_benefit' ( 
  'TREATMENT' smallint(6) NOT NULL,
  'EXECUTION' smallint(6) NOT NULL,
  'AIRLINE' varchar(4) NOT NULL,
  'FUEL_BURN' float unsigned DEFAULT NULL,
  'CONFLICTS' mediumint(7) unsigned DEFAULT NULL,
  'DEPARTURE_DELAY' mediumint(7) DEFAULT NULL,
  'ARRIVAL_DELAY' mediumint(7) DEFAULT NULL,
  'DISTANCE' float unsigned DEFAULT NULL,
  'Q_RECORDS' mediumint(7) unsigned DEFAULT NULL,
  'NON_ZERO_QRECS' mediumint(7) unsigned DEFAULT NULL,
  'CHANGED_QRECS' mediumint(7) unsigned DEFAULT NULL,
  PRIMARY KEY ('TREATMENT', 'EXECUTION', 'AIRLINE'),
  CONSTRAINT 'airline_benefit_ibfk_1' FOREIGN KEY ('TREATMENT', 'EXECUTION') REFERENCES 'executions_sequence' ('TREATMENT', 'EXECUTION') ) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=DYNAMIC COMMENT='Airline-level benefits';
//@40101 SET character_set_client = @saved_cs_client ;

-- Table structure for table 'decision_benefit'

180
DROP TABLE IF EXISTS 'decision_benefit';
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE 'decision_benefit' (  
'TREATMENT' smallint(6) NOT NULL,
'EXECUTION' smallint(6) NOT NULL,
'AIRLINE' varchar(4) NOT NULL,
'TIME' smallint(6) NOT NULL,
'ROUTES_SEQ' mediumint(8) unsigned NOT NULL,
'FUEL_BURN' float DEFAULT NULL,
'CONFLICTS' smallint(6) unsigned DEFAULT NULL,
'DEPARTURE_DELAY' smallint(6) DEFAULT NULL,
'AIRLINE' smallint(6) DEFAULT NULL,
'DISTANCE' float DEFAULT NULL,
'CONGESTED_SECT' float unsigned DEFAULT NULL,
'NUM_SECTORS' tinyint(3) unsigned DEFAULT NULL,
PRIMARY KEY ( 'TREATMENT', 'EXECUTION', 'AIRLINE', 'TIME', 'ROUTES_SEQ' ),
KEY 'decision_benefit_ibfk_2' ( 'ROUTES_SEQ' ),
KEY 'decision_benefit_idx_1' ( 'TREATMENT' ),
KEY 'decision_benefit_idx_2' ( 'EXECUTION' ),
KEY 'decision_benefit_idx_3' ( 'AIRLINE' ),
KEY 'decision_benefit_idx_4' ( 'TIME' ),
CONSTRAINT 'decision_benefit_ibfk_1' FOREIGN KEY ( 'TREATMENT', 'EXECUTION' ) REFERENCES executions_sequence ( 'TREATMENT', 'EXECUTION' ),
CONSTRAINT 'decision_benefit_ibfk_2' FOREIGN KEY ( 'ROUTES_SEQ' ) REFERENCES routes ( 'seq' )
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='The_flight−level_benefits';
/*!40101 SET character_set_client = @saved_cs_client */;

— Table structure for table 'od_pairs'
—

DROP TABLE IF EXISTS 'od_pairs';
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE 'od_pairs' (  
'id' mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
'origin' varchar(15) NOT NULL,
'dest' varchar(15) NOT NULL,
'distance' double DEFAULT NULL COMMENT 'Great Circle Distance in nautical miles',
PRIMARY KEY ( 'id' ),
KEY 'origin' ( 'origin' ),
KEY 'dest' ( 'dest' )
) ENGINE=InnoDB AUTO_INCREMENT=32132 DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

— Table structure for table 'qfunctions'
—

DROP TABLE IF EXISTS 'qfunctions';
/*!40101 SET @saved_cs_client = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `qfunctions` (  
    `TREATMENT` smallint(6) unsigned NOT NULL,  
    `AIRCLine` varchar(4) NOT NULL,  
    `TIME` smallint(6) NOT NULL,  
    `ROUTES_SEQ` mediumint(8) unsigned NOT NULL,  
    `VALUE` float NOT NULL,  
    PRIMARY KEY (`TREATMENT`, `AIRCLine`, `TIME`, `ROUTES_SEQ`),  
    KEY `ROUTES_SEQ` (`ROUTES_SEQ`),  
    CONSTRAINT `qfunctions_ibfk_1` FOREIGN KEY (`ROUTES_SEQ`) REFERENCES `routes` (`seq`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `system_benefit` (  
    `TREATMENT` smallint(6) NOT NULL,  
    `EXECUTION` smallint(6) NOT NULL,  
    `FUEL_BURN` float DEFAULT NULL,  
    `CONFLICTS` mediumint(8) unsigned DEFAULT NULL,  
    `DEPARTURE_DELAY` mediumint(9) DEFAULT NULL,  
    `ARRIVAL_DELAY` mediumint(9) DEFAULT NULL,  
    `PERCENT_TIME_OVERLOADED` float unsigned DEFAULT NULL,  
    `PERCENT_AIRPORT_OVERSCHEDULED` float unsigned DEFAULT NULL,  
    PRIMARY KEY (`TREATMENT`, `EXECUTION`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='The system level benefits';

CREATE TABLE `treatment_params` (  
    `TREATMENT` smallint(6) NOT NULL,  
    `PUSH_BACK_DELAY` float DEFAULT NULL,  
    `SPEED_ERROR` float DEFAULT NULL,  
    `ACCESSIBILITY` varchar(10) DEFAULT NULL,  
    `LATENCY` tinyint(4) DEFAULT NULL,  
    `ACCURACY` float DEFAULT NULL,  
    PRIMARY KEY (`TREATMENT`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 ROW_FORMAT=DYNAMIC;

/*!40014 SET FOREIGN_KEY_CHECKS=OLD_FOREIGN_KEY_CHECKS */
/*!40014 SET UNIQUE_CHECKS=OLD_UNIQUE_CHECKS */
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40101 SET character_set_client = utf8 */;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40101 SET character_set_client = utf8 */;
The following is the script to extract the data to determine convergence to stability of an experiment. The number 3 in the statement section \( treatment = 3 \) in the script must be changed to experiment number to analyze (i.e., 1,2,3,4,5,6).

```sql
select a.treatment, a.execution, tot_flights, decision_flights, epsilon, lambda, tot_qrecs, nzero_qs, ch_q
from
(select treatment, execution, tot_flights, decision_flights, cast(epsilon as decimal(3,2)) as epsilon, lambda
from executions_sequence
where treatment = 3) a,
(select treatment, execution, sum(q_records) as tot_qrecs, sum(non_zero_qrecs) as nzero_qs, sum(changed_qrecs) as ch_q
from airline_benefit
where treatment = 3
group by treatment, execution) b
where a.treatment = b.treatment and a.execution = b.execution
order by execution asc
```

The following script obtains the system-wide metrics. In this script the number 4 is changed to experiment number (called treatment in the database) of experiment being analyzed.

```sql
select treatment, execution, cast(fuel_burn as decimal(10,0)) as fuel_burn, conflicts, departure_delay, arrival_delay, cast(100.0 * percent_time_overloaded as decimal(5,2)) as percent_Ol, cast(100.0 * percent_airport_overscheduled as decimal(5,2)) as percent_OSch
from system_benefit
where treatment = 4
order by treatment, execution;
```

### A.2 Configuration File

The following is an example of a configuration file for a simulation. Similar files must be created for each experiment (the examples correspond to experiment #1) with changes in each line that refers to the experiment number, in the parameters that identify the computer where the database is, the user and password for the database, and in each line that sets
the values of the independent variables.

```bash
# flag to activate the display of messages for the user about current status and configuration of the simulation
verbose=true

# flag to use MASON multi-agent system or not
# Possible values are true and false
usingMason=true

# This is the output DB only
# first driver name (as used by Java). Then computer name (with some prefix identifying the type of DB server). Then user name, then password
# Examples are: com.microsoft.sqlserver.jdbc.SQLServerDriver jdbc:sqlserver://PowerEdge1900 vsnet.gmu.edu dissertation dissertation

# syntax is:
# stochasticity = ['pushback' lambda] ['speed' sigma]
# lambda is the parameter for normalized exponential distribution (0.04 means and average of 25 minutes delay)
# sigma is the parameter for normal distribution (3.33 means +-10knots)
#stochasticity = pushback 25 speed 10

# options are 'all' or 'airline'
accessibility = all

# the value is the number of (executions to delay the information of the past)
# a value of 0 means take from the last result backwards
# a value of 1 means taken from second last result backwards
latency = 0

# the value is the % error of the outputs (from 0 to 1 inclusive)
accuracy = 0.0

# threshold (percentage) to consider differences between metrics significant
comparisonThreshold = 0

landedLimit = 70000

# Description of the output path and files
# do not include the final "/
outputPath = /home/gcalderon/FACET/facet_rexml/user/work
#utilizationFile =utilization_1
#arrivalsFile =arrivals_1
#flightsFile =flights_1
```

184
A.3 Execution of a Simulation

This section describes the shell command used to execute a simulation. The command assumes that simulator has been compiled into a “jar” file named “EvalRoutesSelection.jar”. The file is home directory of the user. FACET is also assumed to be installed in a directory called “FACET” in the home directory of the user. The FACET directory must contain a directory called “facet_rexml” where all the other directories of FACET must exist. This is not the standard structure of later versions of FACET. Therefore, some changes will be
required in this command to reflect the new directory structure of FACET.

The first parameter of the command is the name of the configuration file to be used. It is assumed that the configuration file is in the home directory of the user. The second parameter identification of the experiment (i.e., the experiment number). The third parameter is the number of days to simulate.

### Go to the init directory of FACET

```bash
cd $HOME/FACET/facet_rexml/init

FACET_HOME=$HOME/FACET/facet_rexml
WORKSPACE=$HOME/workspace
```

# With 70,000 flights we need 1.5 GB of memory (at least)

```bash
java -Xms1500m -Xmx1500m -Djava.library.path=$FACET_HOME/bin/ -jar $HOME/EvalRouteSelection.jar
$FACET_HOME $HOME/$1 $2 -repeat $3
```

## A.4 GNUPlot Scripts

This section contains the scripts used to generate charts. The GNU Plot (version 4.4) application was used to generate the charts in the PNG format. The input file is assumed to be a text file, in the comma-separated format, located in the same directory the script file is, and with the filename “compare_exps.csv”.

The following script generates the charts comparing the system-wide metrics of experiments 1 and 2. Similar scripts are used to generate the charts to compare experiments 3 and 1, 4 and 2, 5 and 1, and 6 and 2. The changes are in the variables for the names of the input and output files, the independent variables being compared, the actual values for mean, and standard deviations, and the positions of some labels in the chart.

```bash
# change the default directory to point to the directory where the input files are

```bash
cd 'C:/Users/gcalderon/Documents/Dissertation/Dissertation/analysis/Sys_metrics_compare_charts'/
```

```bash
comparisonLabel = '1,2';
explLabel_0 = 'global';
exp2Label_0 = 'local';
common = '/_real-time_/accurate';
explLabel = explLabel_0 . common;
exp2Label = exp2Label_0 . common;
maxPoints = 80+5;

stability1 = 17;
```
stability2 = 18;

set datafile separator ' ',';

set terminal pngcairo enhanced color font "Times New Roman,12" dashed;
set output '/.../.../.../figs/C_comparisionLabel','.Fuel.png';

# *** These next three are for generating a latex chart
set termoption enhanced

# ***** Set axes labels labels
set ylabel 'Total fuel burn (Million kg/day)' font ',16';
set xlabel 'Simulated day' font ',16';

# ***** Put tics and key
set key bmargin center horizontal
set tics font ',14'

# ***** Set other labels
maxYValue = 304;
minYValue = 298;
unset label
unset arrow
boxLocation = 27.0;
boxTopLocation = maxYValue - (maxYValue - minYValue) * 0.03;
textHeightNumeric = (maxYValue - minYValue) * 0.05;

set label 'Flights/day: 67,753 (52,223 selecting route)' at boxLocation,boxTopLocation -
textHeightNumeric * 0 left font ',13';
set label '{/Symbol l}=0.3' at boxLocation,boxTopLocation - textHeightNumeric * 1 left font ',13';

messageLocation = 302.5;
set arrow from 5,minYValue+0.3 to 1,minYValue;
set label '{/Symbol e}=1.0' at 5.5,minYValue+0.35 left font ',13';
set arrow from 22,minYValue+0.3 to 17,minYValue;
set label '{/Symbol e}=0.2' at 22.5,minYValue+0.35 left font ',13';
set label 'Start of steady state' at 26,messageLocation left font ',13';
set arrow from 25,messageLocation to stability1+0.1,messageLocation = 0.1;
set arrow from stability1,minYValue to stability1,maxYValue nohead lc rgb "light-red" lt 1;
set arrow from 25,messageLocation to stability2+0.1,messageLocation + 0.1;
set arrow from stability2,minYValue to stability2,maxYValue nohead lc rgb "dark-green" lt 3;

mean1 = 300.43
std1 = 0.130

mean2 = 300.27
std2 = 0.109
\[ \text{delta} = '0.159' \]

\[ \text{displace1} = .7 \]
\[ \text{displace2} = 1.3 \]

set arrow from 43, messageLocation - displace1 - 0.02 to 30, mean1 lc rgb "light-red";
set label '{x{0.5-}w{exp1Label0.}w' \[\text{w} \text{printf} \{%.2f\} \text{mean1}\]}w' \[\text{w} \text{printf} \{%.3f\} \text{std1}\] at 43.5, messageLocation - displace1 left font ',13';

set arrow from stability1, mean1 to maxPoints, mean1 nohead lc rgb "light-red" lt 1;

set arrow from 43, messageLocation - displace2 - 0.02 to 30, mean2 lc rgb "dark-green";
set label '{x{0.5-}w{exp2Label0.}w' \[\text{w} \text{printf} \{%.2f\} \text{mean2}\]}w' \[\text{w} \text{printf} \{%.3f\} \text{std2}\] at 43.5, messageLocation - displace2 left font ',13';

set arrow from stability2, mean2 to maxPoints, mean2 nohead lc rgb "dark-green" lt 3;

set arrow from 43, messageLocation - displace1 + 0.4 to 45, messageLocation - displace1 + 4 left font ',13';

unset grid
set xtics 5;
set ytics 1; #0.2;
set format y "%4.1f";

# ************* Display scaled metrics
plot [1:maxPoints] [minYValue:maxYValue] 'compare_exp_.comparisonLabel.' csv using 1:(\$2 / 1.0e+6) with lines lt 1 lc rgb "light-red" lw 2 title 'Information_is_.exp1Label' \ 'compare_exp_.comparisonLabel.' csv using 1:(\$8 / 1.0e+6) with lines lt 3 lc rgb "dark-green" lw 2 title 'Information_is_.exp2Label';

# ************* plot for conflicts

set output '...' / '.../.../.../figs/C_' comparisonLabel '._Conflicts.png'

set ylabel 'Total conflicts (Thousands/day)' font ',16';
maxYValue = 70;
minYValue = 55;
unset label
unset arrow
boxLocation = 27.0;
boxTopLocation = maxYValue - (maxYValue - minYValue) * 0.03;
textHeightNumeric = (maxYValue - minYValue) * 0.05;

set label '{Flights_/_day:67,753(52,223_selecting_route)}' at boxLocation, boxTopLocation - textHeightNumeric * 0 left font ',13';
set label '{/Symbol_1}=0.3' at boxLocation, boxTopLocation - textHeightNumeric * 1 left font ',13';

messageLocation = 67; #65.5;
```plaintext
set arrow from 5, minYValue+0.5 to 1, minYValue;
set label '{\text{
\textbf{e}}}' at 5.5, minYValue+0.7 left font ',13';
set arrow from 22, minYValue+0.5 to 17, minYValue;
set label '{\text{
\textbf{e}}}' at 22.5, minYValue+0.7 left font ',13';
set label 'Start of steady state' at 26, messageLocation left font ',13';
set arrow from stability1, minYValue to stability1, maxYValue nohead lc rgb "light-red" lt 1
set arrow from 25, messageLocation to stability1 +0.1, messageLocation -0.2;
set arrow from stability1, minYValue to stability1, maxYValue nohead lc rgb "dark-green" lt 3;
mean1 = 61.1389
std1 = 0.366
mean2 = 62.8338
std2 = 0.191
delta = '-1.694'
displace1 = 2.5;
displace2 = 1.5;
set arrow from 43, messageLocation -displace1 -0.02 to 30, mean1 nohead lc rgb "light-red";
set label '{\text{
\textbf{x}}}' \(\text{'0.5-'\textbf{\text{}{}}} \text{}\text{.exp1Label}{}\text{}\text{}\text{.exp1Label}{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\text{}\t
maxYValue = 140;
minYValue = 100;

unset label
unset arrow
boxLocation = 27.0;
boxTopLocation = maxYValue - (maxYValue - minYValue) * 0.03;
textHeightNumeric = (maxYValue - minYValue) * 0.05;

set label 'Flights/day: 67,753(52,223 selecting route)' at boxLocation, boxTopLocation - textHeightNumeric * 0 left font ',13';
set label '{/Symbol l}=0.3' at boxLocation, boxTopLocation - textHeightNumeric * 1 left font ',13';

messageLocation = 133; #136.5;

set arrow from 5, minYValue + 1.5 to 1, minYValue;
set label '{/Symbol e} = 1.0' at 5.5, minYValue + 1.7 left font ',13';
set arrow from 22, minYValue + 1.5 to 17, minYValue;
set label '{/Symbol e} = 0.2' at 22.5, minYValue + 1.7 left font ',13';

set label 'Start of steady state' at 26, messageLocation left font ',13';
set arrow from 25, messageLocation to stability1 + 0.1, messageLocation = 0.5;
set arrow from stability1, minYValue to stability1, maxYValue nohead lc rgb "light-red" lt 1
set arrow from 25, messageLocation to stability2 + 0.1, messageLocation + 0.5;
set arrow from stability2, minYValue to stability2, maxYValue nohead lc rgb "dark-green" lt 3;

mean1 = 119.02
std1 = 2.48

mean2 = 118.68
std2 = 2.93

delta = '0.340'

displace1 = 6
displace2 = 8

set arrow from 43, messageLocation - displace1 - 0.5 to 30, mean1 lc rgb "light-red";
set label '{x{0.5-} \{.exp1Label_0 .\} \{.exp1Label_0 .\} \{.exp1Label_0 .\} \{.exp1Label_0 .\}} = \{.exp1Label_0 .\} = \{.exp1Label_0 .\} = \{.exp1Label_0 .\} = \{.exp1Label_0 .\};
\text{printf("%.3f", std1)} at 43.5, messageLocation - displace1 left font ',13';

set arrow from stability1, mean1 to maxPoints, mean1 nohead lc rgb "light-red" lt 1;
set arrow from 43, messageLocation - displace2 - 0.5 to 30, mean2 lc rgb "dark-green";
set label '{x{0.5-} \{.exp2Label_0 .\} \{.exp2Label_0 .\} \{.exp2Label_0 .\} \{.exp2Label_0 .\} \{.exp2Label_0 .\} \{.exp2Label_0 .\} \{.exp2Label_0 .\} \{.exp2Label_0 .\}} = \{.exp2Label_0 .\} = \{.exp2Label_0 .\} = \{.exp2Label_0 .\} = \{.exp2Label_0 .\} = \{.exp2Label_0 .\} = \{.exp2Label_0 .\} = \{.exp2Label_0 .\};
\text{printf("%.3f", std2)} at 43.5, messageLocation - displace2 left font ',13';

set arrow from stability2, mean2 to maxPoints, mean2 nohead lc rgb "dark-green" lt 3;
set label "x{0.5−} irresistible {0.5−} unable {0.5−} delta at 45,
messageLocation = displace1 + 4.5 left font ',13';
unset grid
set ytics
set format y "%3.0f"
plot [1:maxPoints] [minYValue:maxYValue] 'compare_exp_.comparisonLabel_.csv' using 1:($4 / 1e+3) with lines lt 1 lc rgb "light-red" lw 2 title 'Information is _1_.exp1Label_ .
'compare_exp_.comparisonLabel_.csv' using 1:($10 / 1e+3) with lines lt 3 lc rgb "dark-green" lw 2 title 'Information is _2_.exp2Label_ .

# ******* plot for arrival delay

set output '../..../figs/C'.comparisonLabel_ _ArrDelay_.png'

set ylabel 'Total arrival delay (Thousand minutes / day)' font ',16';
maxYValue = 590;
minYValue = 550;
unset label
unset arrow
boxLocation = 27.0;
boxTopLocation = maxYValue - (maxYValue - minYValue) * 0.03;
textHeightNumeric = (maxYValue - minYValue) * 0.05;
set label 'Flights/day: 67,753 (52,233 selecting route)' at boxLocation, boxTopLocation -
textHeightNumeric * 0 left font ',13';
set label '{/Symbol} = 0.3' at boxLocation, boxTopLocation = textHeightNumeric * 1 left font ',13';
messageLocation = 580;
set arrow from 5, minYValue + 1.5 to 1, minYValue;
set label '{/Symbol} = 1.0' at 5.5, minYValue + 2 left font ',13';
set arrow from 22, minYValue + 1.5 to 17, minYValue;
set label '{/Symbol} = 0.2' at 22.5, minYValue + 2 left font ',13';
set label 'Start of steady state' at 26, messageLocation left font ',13';
set arrow from 25, messageLocation to stability1 + 0.1, messageLocation - 0.5;
set arrow from stability1, minYValue to stability1, maxYValue nohead lc rgb "light-red" lt 1
set arrow from 25, messageLocation to stability2 + 0.1, messageLocation + 0.5;
set arrow from stability2, minYValue to stability2, maxYValue nohead lc rgb "dark-green" lt 3;

mean1 = 563.94
std1 = 1.65

mean2 = 562.84
std2 = 2.08

delta = '1.104'
displace1 = 6
displace2 = 8

set arrow from 46, messageLocation−displace1−1.5 to 35, mean1 lc rgb "light−red";

set label ’\text{x}(0.5−)_{\text{\scriptsize exp1Label}_0} \text{, \scriptsize gprintf("\%f", mean1)} \text{, \scriptsize gprintf("\%f", std1) at 46.5, messageLocation\−displace1\−0.5 left font \',13}’;

set arrow from stability1, mean1 to maxPoints, mean1 nohead lc rgb "light−red" lt 1;

set arrow from 46, messageLocation−displace2−1.5 to 38, mean2 lc rgb "dark−green";

set label ’\text{x}(0.5−)_{\text{\scriptsize exp2Label}_0} \text{, \scriptsize gprintf("\%f", mean2)} \text{, \scriptsize gprintf("\%f", std2) at 46.5, messageLocation\−displace2\−0.5 left font \',13}’;

set arrow from stability2, mean2 to maxPoints, mean2 nohead lc rgb "dark−green" lt 3;

set label ’\text{x}(0.5−)_{\text{\scriptsize exp1Label}_0} \text{, \scriptsize gprintf("\%f", mean1)} \text{, \scriptsize gprintf("\%f", std1) at 48, messageLocation\−displace1+4 left font \',13}’;

unset grid

set ytics 4;
set format y "\%3.0f";

plot [1:maxPoints] [minYValue:maxYValue] ’compare_exp_.comparisonLabel_.csv’ using 1:($5 / 1e+3) with lines lt 1 lc rgb "light−red" lw 2 title ’Information \_{\text{\scriptsize is \text{\scriptsize compLabel}_0}}’ \text{\scriptsize \text{\scriptsize \text{\scriptsize using \\text{\scriptsize 1:($11 / 1e+3) with lines lt 3 lc rgb "dark−green" lw 2 title \text{\scriptsize Information \_{\text{\scriptsize is \text{\scriptsize compLabel}_0}}’;}}}

# *********** plot for % OL

set output ’\text{\scriptsize \text{\scriptsize /\scriptsize figs/C}_0\text{\scriptsize .comparisonLabel}_0\text{\scriptsize .PerOL.png}’

set ylabel ’% OL’ font ',16’;

maxYValue = 80;
minYValue = 70;

unset label

unset arrow

boxLocation = 27.0;
boxTopLocation = maxYValue − (maxYValue − minYValue) * 0.03;
textHeightNumeric = (maxYValue − minYValue) * 0.05;

set label ’\text{\text{\scriptsize Flights/day: \text{\scriptsize 67,753}}_{\text{\scriptsize (52,223 selecting route)}}\text{\scriptsize at boxLocation, boxTopLocation = textHeightNumeric * 0 left font \',13}\text{\scriptsize ;}}

set label ’\text{/Symbol}_{\text{\scriptsize 0.3}}’ at boxLocation, boxTopLocation = textHeightNumeric * 1 left font \',13\text{\scriptsize ;}}

messageLocation = 78;

set arrow from 5, minYValue+0.4 to 1, minYValue;

set label ’\text{/Symbol}_{\text{\scriptsize 1.0}}’ at 5.5, minYValue+0.5 left font \',13\text{\scriptsize ;}}
mean1 = 73.38
std1 = 0.464
mean2 = 72.99
std2 = 0.385
delta = '0.39'
displace1 = 1.5;
displace2 = 2.0;

maxYValue = 12;
minYValue = 10;
A.5 R Scripts

This section contains the scripts used to perform the normality tests, variance equality tests, mean equality tests, mean difference tests, and variance difference tests. The R (version 2.11.1) application was used for these tasks.

A.5.1 Script to Test for Normality of the Datasets

The tests for normality are performed using the Shapiro test provided by R. The input file is assumed to be a text file, in the comma-separated format without headers, located in the same directory the script file is, and with the filename “compare_exps.csv”.

```r
# Testing for normality
a <- read.table("C:\Users\gcalderon\Documents\Dissertation\analysis\sys_metrics\compare_charts\compare_exps.csv", header = FALSE, sep = ",");
row <- nrow(a)
# separator for display purposes
separator = "\n"
# first data point to consider in the analysis for each experiment
base_r <- c(17,18,21,26,23,32)
# iterate over all experiments
# apply functions to the columns of the matrix
for (experiment in c(1,2,3,4,5,6)) {
  cat("nExperiment",experiment,"\n")
  cat("Order_of_results_is_fuel_burn\_conflicts\_departure\_delay\_arrival\_delay\_\%OL\_\%OSch\n")

  sel_col <- (experiment - 1) * 6 + 2;
i <- e(base_r[experiment],row)
f <- e(sel_col,sel_col+5)
  cat("Descriptive\_statistics\_and\_normality\_test\n")
  cat("min\_average\_median\_max\_std\_W\_statistic\_p\_value\n")
}````
apply(a[i,f],2, function(x) {
  s <- subset(x, !is.na(x));
  cat(min(s),mean(s),median(s),max(s),sd(s),sep = separator);
  cat(separator);
  s <- shapiro.test(subset(x, !is.na(x)));
  cat(s[[1]]$statistic, s[[3]]$p.value, "n", sep = separator));
})

A.5.2 Script to Test for Equality of the Variances

The tests for equality of variances are performed using the F-test and the Levene-test functions provided by R. The input file is assumed to be a text file, in the comma-separated format without headers, located in the same directory the script file is, and with the filename "compare_exps.csv".

```r
# Testing for equality of the variances
# Loading package to have Levene's test available
require(car, quietly = TRUE);

myLeveneTest = function(a1, a2, l1, l2) leveneTest(c(a1, a2), factor(c(seq(length=l1, from="A", to="A"), seq(length=l2, from="B", to="B"))));

a <- read.table(paste("C:\Users\gcalderon\Documents\Dissertation\analysis\sys_metrics\compare_charts\\correction\compare_exps.csv", sep=""), header = FALSE, sep = ",");

# first data point to consider in the analysis for each experiment
base <- c(17,18,21,26,23,32);
comparisons <- matrix(c(1,2,3,1,4,2,5,1,6,2), nrow=2);
metrics <- c("Total_fuel_burn","Total_airborne_conflicts","Total_departure_delay","Total_arrival_delay","%OL","%Osch");

# separator for display purposes
separator = ","

# recording normality results for each metric (horizontal) or each experiment (vertical)
normals <- matrix(c(1,1,0,0,1,0), c(1,1,0,1,1,0), c(1,1,0,1,1,0), c(1,1,0,1,1,0), c(1,1,0,1,1,0), c(1,1,0,0,1,0)), ncol=6);

for(comparison in c(1,2,3,4,5)) {
  cat("Comparison", comparison);
  expA <- comparisons[,comparison][1]
  expB <- comparisons[,comparison][2]
  # additional code to test normality...
}
```
A.5.3 Script to Test the Hypotheses for the Difference of Means

The tests for difference of means are performed using two variations of the t-test and the Wilcoxon-test provided by R. The input file is assumed to be a text file, in the comma-separated format without headers, located in the same directory the script file is, and with the filename “compare_exps.csv”.

```r
# Testing for difference in the means
a <- read.table(paste("C:\Users\gcalderon\Documents\Dissertation\Dissertation\analysis\sys_metrics_compare_charts\correction\compare_exps.csv",sep=""), header = FALSE, sep = ",");

# first data point to consider in the analysis for each experiment
base <- c(17,18,21,26,32)
comparisons <- matrix(c(1,2,3,1,4,2,5,1,6,2),nrow=2);

metrics <- c("Total_fuel_burn","Total_airborne_conflicts","Total_departure_delay","Total_arrival_delay","%OL","%Osch");

# separator for display purposes
separator = ","
```

# this is the first data point to consider for the analysis
base_r1 <- base[expA]
base_r2 <- base[expB]

i1 <- c(base_r1:nrow(a))
i2 <- c(base_r2:nrow(a))

typ <- normals[,expA] * normals[,expB];

cat("Testing for equality of variances\n");
cat("Metric p-value ratio CI(min) CI(max) statistic \n");
for(metric in c(1,2,3,4,5,6)) {
  sel_col1 <- (expA - 1) * 6 + metric + 1;
  sel_col2 <- (expB - 1) * 6 + metric + 1;

  if (typ[metric]) {
    s <- var.test(a[i1,sel_col1], a[i2,sel_col2], alternative = "two.side");
  } else {
    s <- myLeveneTest(a[i1, sel_col1], a[i2, sel_col2], length(i1), length(i2));
    cat(metrics[metric], s[3]$'Pr(>F)'[1],"N/A","N/A",s[2]$'F_value'[1],"\n",sep = separator);
  }
}
```
# recording normality results for each metric (horizontal) or each experiment (vertical)

```r
normals <- matrix(
  c(c(1,1,0,0,1,0),
    c(1,1,0,1,1,0),
    c(0,1,0,1,1,0),
    c(1,1,0,1,1,0),
    c(1,1,0,1,1,0),
    c(1,1,0,1,1,0)),nrow=6);

equals <- matrix(
  c(c(1,0,1,0,1,1),
    c(1,1,0,1,1,1),
    c(1,1,0,0,1,1),
    c(1,0,0,0,0,1),
    c(0,0,1,1,1,1)),nrow=6);

sides <- c("less","greater","greater","greater","greater");

for(comparison in c(1,2,3,4,5)) {
  cat("Comparison ", comparisons[,] , comparison);
  expA <- comparisons [,comparison][1]
  expB <- comparisons [,comparison][2]

  # this is the first data point to consider for the analysis
  base_r1 <- base[expA]
  base_r2 <- base[expB]
  i1 <- c(base_r1:nrow(a))
  i2 <- c(base_r2:nrow(a))

  typ <- normals[,expA] * normals[,expB];

  alternativeStr = sides[comparison];

  cat("Testing for equality of variances\n");
  cat("Metric p-value \n confident interval\n statistic\n");
  for(metric in c(1,2,3,4,5,6)) {
    sel_col1 <- (expA - 1) * 6 + metric + 1;
    sel_col2 <- (expB - 1) * 6 + metric + 1;
    if (typ[metric]) {
      s <- t.test(a[i1,sel_col1], a[i2,sel_col2], alternative = alternativeStr, paired = FALSE, var. equal = equals[metric,comparison])
    } else {
      s <- wilcox.test(a[i1,sel_col1], a[i2,sel_col2], conf.int = TRUE, alternative = alternativeStr, paired = FALSE)
    }
  }
}
```
A.5.4 Script to Test the Hypotheses for the Ratio of Variances

The tests for difference of means are performed using two variations of the t-test and the Wilcoxon-test provided by R. The input file is assumed to be a text file, in the comma-separated format without headers, located in the same directory the script file is, and with the filename “compare_exps.csv”.

```r
# Testing for difference in variances
a <- read.table(paste("C:\\Users\\gcalderon\\Documents\\Dissertation\\Dissertation\\analysis\\sys\metrics_compare_charts\\correction\\compare_exps.csv",sep=""), header = FALSE, sep = " ",)

# first data point to consider in the analysis for each experiment
base <- c(17,18,21,26,23,32)
comparisons <- matrix(c(1,2,3,1,4,2,5,1,6,2),nrow=2);
metrics <- c("Total_fuel_burn","Total_airborne_conflicts","Total_departure_delay","Total_arrival_delay","%OL","%Osch");

# separator for display purposes
separator = " 
"
sides <- c("less","greater","greater","greater","greater");

for(comparison in c(1,2,3,4,5)) {
  cat("Comparison",comparisons[,comparison]);
  expA <- comparisons[,comparison][1]
  expB <- comparisons[,comparison][2]

  # this is the first data point to consider for the analysis
  base_r1 <- base[expA]
  base_r2 <- base[expB]

  i1 <- c(base_r1:nrow(a))
  i2 <- c(base_r2:nrow(a))

  cat("Testing for equality of variances\n");
  cat("Metric p-value CI(min) CI(max) statistic\n");
  for(metric in c(1,2,3,4,5,6)) {
    sel_col1 <- (expA - 1) * 6 + metric + 1;
    sel_col2 <- (expB - 1) * 6 + metric + 1;
    s <- var.test(a[i1,sel_col1], a[i2,sel_col2], alternative = sides[comparison]);
  }
}
```
Appendix B: Output Files of the Experiments

These sections describe the output files optionally generated by the simulations. The generation of these files is controlled by the configuration file for the simulation.

B.1 The Flight Records File

The flight records file is a comma-separated text file. It contains flight-by-flight information about all the flights in the simulation. Each line of the file describes a single flight in the simulation. The format is described in Grammar B.1:

<table>
<thead>
<tr>
<th>Grammar B.1 BNF-like description of the flight records file</th>
</tr>
</thead>
<tbody>
<tr>
<td>(file) → (record)^+</td>
</tr>
<tr>
<td>(record) → (flight_id) (centers) (sectors) (distance) (actual_landing_time)</td>
</tr>
<tr>
<td>(destination_airport) (scheduled_takeoff_time) (scheduled_landing_time) (fuel_burn)</td>
</tr>
<tr>
<td>(flight_id) → (alpha)^+ (digit)^+</td>
</tr>
<tr>
<td>(centers) → 0</td>
</tr>
<tr>
<td>(sectors) → 0</td>
</tr>
<tr>
<td>(distance) → (digit)^+ (., (digit)^+)?</td>
</tr>
<tr>
<td>(actual_takeoff_time) → (-1</td>
</tr>
<tr>
<td>(actual_landing_time) → (-1</td>
</tr>
<tr>
<td>(destination_airport) → (alpha) ((alpha)</td>
</tr>
<tr>
<td>(scheduled_takeoff_time) → (-1</td>
</tr>
<tr>
<td>(scheduled_landing_time) → (-1</td>
</tr>
<tr>
<td>(fuel_burn) → (digit)^+ (., (digit)^+)?</td>
</tr>
</tbody>
</table>

Each, but the last, element of a line is separated from the contiguous elements by a comma. Lines are separated by a line feed (i.e., UNIX style). The number of centers and sectors are set to 0 in all the experiments of this dissertation. A value of -1 in a numeric field means that there was no information. The distance is given in nautical miles. The times are given in milliseconds from January 1 1970 00:00:00 GMT. The fuel burned is given in gallons. The FLIGHT_ID is the flight number the passengers usually see printed in their tickets. The following is a fragment of a file like this.

200
Example B.1 Fragment of a flight records file
ASA162,0,0,56.9533117989348,1155775980000,1155776700000,KANC,-1,1155778278558,0.0
EZY3230,0,0,0,1155778920000,1155778980000,EGKK,-1,1155785066411,100.54
N684MC,0,0,0,1155783060000,1155783120000,KBFI,-1,1155802148018,200.87
CAL31,0,0,2826.023530993964,1155774480000,1155795600000,RCTP,-1,1155813066540,365.23
MYT413,0,0,468.97354613492234,1155792000000,1155795840000,LEGE,-1,1155796499058,766.54
ALK505,0,0,2360.1697963040892,1155775200000,1155795900000,EGLL,-1,1155816267562,876.32
FLC70,0,0,3.1099024356070797,1155796080000,1155796200000,KACY,-1,1155798859547,1200.44

This file allows the computation of distance flown, departure and arrival delays, flight
duration, destination airport congestion, and fuel burn for both the system and the airlines.

B.2 The Sector Utilization File

The sector utilization file records the total number of aircraft flying in a sector at each
minute. The file is comma-separated and its structure is described by Grammar B.2.

Grammar B.2 BNF-like description of the sector utilization file

\[(file) \rightarrow (record)^+ \]
\[(record) \rightarrow (time\_stamp) (center) (level) (sector\_name) (capacity) (num\_aircraft) \]
\[(time\_stamp) \rightarrow (digit)^{13} \]
\[(center) \rightarrow (alpha)^+ ((alpha)|(digit))^+ \]
\[(level) \rightarrow LOW|HIGH|SUPER \]
\[(sector\_name) \rightarrow (alpha)^+ ((alpha)|(digit))^+ \]
\[(capacity) \rightarrow (digit)^{1-4} \]
\[(num\_aircraft) \rightarrow (digit)^{1-4} \]

The file contains records for sectors that contained one or more aircraft in that time
step, empty sectors are not reported. The time stamp is the number of milliseconds from
Jan 1, 1970 00:00:00 GMT. The capacity is given in number of aircraft. A fragment of one
file is shown by Example B.2.
Example B.2  Fragment of a sector utilization file

1155774480000,Anchorage,High,ZAN11,21,1
1155774540000,Anchorage,High,ZAN11,21,1
1155779160000,Miami,High,ZMA40,18,10
1155844860000,Edmonton,High,YEGEN,15,13

B.3 The Airport Arrivals File

The airport arrivals file is a comma-separated text file derived from the flight records file. It is obtained by a parsing process that filters information and summarizes data. The structure of the file is described by Grammar B.3.

Grammar B.3  BNF-like description of the airport arrivals file.

\[
\langle\text{file}\rangle \rightarrow (\langle\text{airport}\rangle \langle\text{time}\_\text{stamp}\rangle \langle\text{arrivals}\_\text{fp}\rangle \langle\text{arrivals}\_\text{direct}\rangle)^{+} \\
\langle\text{airport}\rangle \rightarrow \langle\alpha\rangle^{+}(\langle\alpha\rangle | \langle\text{digit}\rangle)^{+} \\
\langle\text{time}\_\text{stamp}\rangle \rightarrow \langle\text{digit}\rangle^{13} \\
\langle\text{arrivals}\_\text{fp}\rangle \rightarrow \langle\text{digit}\rangle^{+} \\
\langle\text{arrivals}\_\text{direct}\rangle \rightarrow \langle\text{digit}\rangle^{+}
\]

The airport element is the code of the destination airport (usually ICAO code). The time stamp marks the start of a time period. It can appear several times in the file associated to different airports. The time period is defined by a parameter for the conversion, and its value represents a number of minutes, e.g., 15, 30, or 60. The next two numerical values count the number of arrivals at that airport and time period when flights use airway routes (the first value) and direct routes (the second value).

For the purpose of this dissertation only airports which code starts with the letters “K” or “P” (e.g., “KJFK”, “PANC”, “PHNL”) are considered and reported. The rest are ignored.
Appendix C: Java Source Code

This section contains sections of the Java source code of the Main Application component of the architecture. The full complete source code is available in the website of the Center for Air Transportation Systems Research (http://catsr.ite.gmu.edu/).

C.1 FACETExperiment Class

This section contains sections of the source code from the class FACETExperiment.

The following code is a section of the method initialize. The method starts FACET through the API, calls the method that performs the flightplan route selection process and configures the display of FACET.

```java
/* Start up FACET by giving it the location of a FACET installation */
if (facet_api == null) {
    facet_api = FACETServerAPI.getInstance();
    facet_api.loadFACET( FACET_homeDirectory );
} else {
    facet_api.getSim().terminate(); // Terminate simulation just in case
}

aircraft = new ProtectedAircraftInterface(facet_api.getAircraft());
utils = facet_api.getUtils();

// Moved here to have valid interfaces during the pre-processing
readInputParams();
preProcessInput();

// Clean the display
facet_api.getGUI().eraseAllObjects();

/* Do some fun GUI stuff */
// First display the center only (low and high)
facet_api.getGUI().showBoundariesUSA( false, false, false, true, true );
facet_api.getGUI().showUSBoundary( true );

// Look up the location of the OEP-35 airports
int[] drawColor = facet_api.getGUI().getDrawColor();
double[] loc = null;
String[] OEP35 = {"KATL","KBOS","KBWI","KCLE","KCVG","KDCA","KDFW","KDEN","KDFW","KEWR","KDFW","KDFW","KLAX","KLDN","KMAH","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM","KMEM"};
```
The following code is the method `preProcessInput`, called from the method `initialize` described above. The code reads the input file, performs the route selection for all the airlines and writes a new file with the selected routes. The new input file will be read later by FACET. This is an effective way to perform pre-departure flightplan route selection without having to directly change flights in FACET or create them during the simulation.
path = file.getParent();
child = inputFilename.substring(inputFilename.indexOf(path) + path.length() + 1,
inputFilename.indexOf(path) + path.length() + 11);

// Extracting the year, month, and day of the simulation from the filename of the input
// file (assuming a standard format for the filename)
SimpleDateFormat form = new SimpleDateFormat("yyyy MMM dd");
Date date = form.parse(child);
child = inputFilename.substring(inputFilename.indexOf(path) + path.length() + 1);

// First find the time limits
parser = new PNPParse(new BufferedReader(new InputStreamReader(new DataInputStream(new FileInputStream(inputFilename))))), date.getTime());
if (parser.hasMoreTokens())
    parser.skipLines(1); // Skip the headings
break;
break;
}
break;
case DB:
    // TODO this is not consistent with the load method in FACETAgent
inputFilename = props.getProperty("outputPath");
file = new File(inputFilename);
path = file.getAbsolutePath();
child = "";

// The format of date and time is YYYY-MM-DD (e.g. 2007−07−27) and HH:MM:SS (e.g. 00:00:00)
String dateTime = getInPrams().getDbStartDate() + " " + getInPrams().getDbStartTime();
parser = new DBParser(inputDatabase,dateTime);
break;
}

// TODO check if this is OK when using named pipes
// Delete the dummy input file if it already exists
file = new File(path, child + ".dummy" + TREATMENT); // Include the treatment number for safety (not collide with other instances)
if (file.exists()) {
    file.delete();
}

if (routes == null) {
    if (isVerbose)
        System.out.println("Loading routes");
    routes = new RoutesCollection(database);
} else {
    if (isVerbose)
        System.out.println("Skipping routes loading process");
}
if (isVerbose)
System.out.println("Total routes: "+routes.size());
database.executeUpdate("update executions_sequence set tot_routes =" + routes.size() +
    " where treatment =" + TREATMENT + " and execution =" + execution);
// Build a reasonable big hash table for the raw flights (the key is the flight number....
// assuming only 24 hr will be simulated)
int size_limit = 20000;
String propValue = props.getProperty("landedLimit");
if (propValue != null && !propValue.isEmpty()) {
    size_limit = Integer.parseInt(propValue);
}
rawListFlights = new java.util.ArrayList<FlightRecord>(size_limit);

// Prepare to use push back delays
ProbDistribution pushBackDistrib = createPushbackDistribution(_pushbackDelay, false);

java.util.HashMap<Long, java.util.ArrayList<FlightRecord>> tempNewFlights =
    new java.util.HashMap<Long, java.util.ArrayList<FlightRecord>>();

// Read the input using the parser and transform it as it is convenient
int flightsRead = 0;
int flightsChoosingAction = 0;

PrintStream myOut = new java.io.PrintStream(new java.io.BufferedOutputStream(
    new java.io.FileOutputStream(file)));
FlightRecord record = null;
Route route = null;
while (parser != null && parser.hasMoreTokens()) {
    // Stop when there no more tokens (end of file) or the last record could not be inserted (too early)
    record = parser.readRecord();
    if (record == null) // Ignore empty groups (perhaps because there were syntax errors in the input)
        continue;
    flightsRead++;
    //Find the airline and include it in the simulation if not there already
    String airlineName = record.getAirlineName();
    if (airlineName == null) {
        System.err.println("Too short name of flight:
    } else {
        DecisionMakingAirline airline = (DecisionMakingAirline)recordAirlineAgent(airlineName);
        route = routes.makeSureRouteIsThere(record.origin, record.destination, Route.
            removeSuffix(record.flightPlan), database, is_verbose);
        // Choose a route before take off
        short minutes = TimeMapper.getMinutesOfDay(record.sch_dep_time);
        if (using_Adaptation) {
            if (((record.origin.length() == 3 || AirportCodeConverter.isUSAirport(record.origin)
                    ) &&
                !route.isLoop()) // US departures only, and no route selection for loops (usually training flights)
                ProtectedAirportInterface airports = getAirportInterface();
            double[] coords = airports.getLocation(AirportCodeConverter.from3to4(record.origin));
            if (coords != null) {
                tempNewFlights.put(record.origin, new ArrayList<FlightRecord>(
                    1));
onf
            } else { // TODO
                continue;
            }
            tempNewFlights.put(record.origin, new ArrayList<FlightRecord>(
                1));
            myOut.println("Warning: Unhandled flight: ");
            continue;
        } else { // TODO
            continue;
        }
    }
}
double lat = Coordinates.normalizeDeg(Coordinates.convertInt2Double(record.latitude));
double lon = Coordinates.normalizeDeg(Coordinates.convertInt2Double(-record.longitude));
double dist = util.getGCDistanceNM(lat, lon, coords[0], coords[1]);
if (dist <= distanceToAirportThreshold) { // The flight is still close to the airport (aircraft do not reach their cruise altitude until about 30 miles from the airport)
    route = airline.chooseAction(minutes, route);// this
    // Inserting push-back delay after route selection...
    record.pushBackDelay = (int)pushBackDistrib.next();
    flightsChoosingAction++;
}
} }
else {
    if (record.origin.length() == 3 || AirportCodeConverter.is4USAirport(record.origin)) {
        // US departures only
        airline.recordAction(minutes, route);
    }
}
record.flightPlan = route.getRoute();
rawListFlights.add(record);

// TODO here I am putting all flights (even if a cannot identify their airline in the dummy input file), but NOT in rawListFlights

// Save the modified record new route, new actual departure (because of the push-back delay)
// Notice that I am not changing the sch_dep_time of the record, but I am including the push-back delay in the input file
// Since I already have the record in rawListFlights, then I am able to compute the delay correctly in later stages of the simulation
// because actual_dep = sch_dep + pushback_delay + ground_delay
long myTime = record.sch_dep_time + record.pushBackDelay * 60000;
java.util.ArrayList<FlightRecord> tempListRecord = tempNewFlights.get(myTime);
if (tempListRecord == null) {
    tempListRecord = new java.util.ArrayList<FlightRecord>();
    tempNewFlights.put(myTime, tempListRecord);
}
tempListRecord.add(record);
}

Long[] myArray = new Long[tempNewFlights.size()];
tempNewFlights.keySet().toArray(myArray);
Arrays.sort(myArray);
// Now I have to write the file as a TRX
for(Long key : myArray) {
    myOut.println("TRACK\TIME=" + key / 1000);
    for(FlightRecord myRecord : tempNewFlights.get(key)) {
        myOut.println(myRecord.toString());
    }
    myOut.println();
}
myOut.println();
}
myOut.close();

if (isVerbose) {
    System.out.print("Total flights read from input: ") ;
    System.out.println(flightsRead);
    System.out.print(" Flights choosing route: ") ;
    System.out.println(flightsChoosingAction);
    System.out.print(" Total airlines: ") ;
    System.out.println(airlines.size());
}
database.executeUpdate(" update executions_sequence set tot_flights = " + flightsRead + " ,
    decision_flights = " + flightsChoosingAction + " ,
    tot_airlines = " + airlines.size() + " ,
    where_treatment = " + TREATMENT + " and execution = " + execution);
}
catch (java.io.FileNotFoundException e) {
    e.printStackTrace(System.err);
}
catch (Exception e) {
    e.printStackTrace(System.err);
}
}

The following code is the method start (required by MASON), which initializes the simulation and creates the FACETAgent.

```java
/*
 * This method is executed once per simulation, before any other agent's method is called.
 * It is mainly used to initialize the simulation.
 */
@Override
public void start() {
    if (usingMASON) {
        super.start();
        initialize(FACETExperiment.FACET_HOME_DIR, FACETExperiment.CONFIG_FILENAME);  //I must do this here so that job() has the correct value
        //TODO: make this an option only if we want the airport capacities to be restricted
        if (job() == 0) {  //This is a provision for the case in which we are repeating the same treatment several times (executions)
            System.err.println("Please, load the airport restrictions and set the output file for the attributes (if in use) ");
        } else if (isVerbose) {
            System.out.println("Airport constraints already loaded or not in use. Skipping reload ");
        }
        if (usingMASON) {
            schedule.scheduleRepeating(0, 0, new FACETAgent(this, inputParams));
        }
    }
}
```
Here we're going to pick whether or not to use Diffuser (the default) or if we're really going for the gusto and have multiple processors on our computer, we can use our multithreaded super-neato ThreadedDiffuser! On a Power Mac G5 with two processors, we get almost a 90% speedup in the underlying model because so much time is spent in the Diffuser.

// FlightSchedule the diffuser to happen after the heatbugs
if (availableProcessors() > 1) {
    System.out.println("Multi-processor mode<" + availableProcessors() + "=" + processors >");
    schedule.scheduleRepeating(schedule.EPOCH, 2, new sim.app.heatbugs.ThreadedDiffuser(), 1);
} else {
    System.out.println("Single-processor mode");
    schedule.scheduleRepeating(schedule.EPOCH, 2, new Diffuser(), 1);
}

C.2 FACETAgent Class

This section includes the source code for the FACETAgent class.

The following code is the method step (required by MASON), which checks if the simulation is finished, registers new flight agents, counts sector utilization, and commands FACET to perform the next step of the simulation.

```java
/** (non-Javadoc)
 * @see sim.engine.Steppable#step (sim.engine.SimState)
 */
@Override
public void step(SimState state) {
    FACETExperiment experiment = (FACETExperiment) state;
    // TODO there are some more things to do here
    try {
        if (experiment.moreSteps()) {
            registerNewAgents(experiment);
            // TODO this can be done in a separate thread (concurrent with the sequence
            // registerNewAgent, processFlights)
            experiment.countSectorUtilization(false);
            // TODO check if this is going to cause any problem because it removes flights from the
            // list
            if (!FACETExperiment(using_MASON)
                experiment.processFlights();
                experiment.step(); // One more step
            ) else if (FACETExperiment(using_MASON)
                // TODO use stoppable here too as I did with the flight agents
                experiment.kill(); // Hard way to finish the simulation
```
C.3 Airline Classes

The following code includes some static variables and the methods perform adaptation, i.e. `adapt`, `computeReward`.

```java
private static final String ADAPT_SQL_GLOBAL = "select d.fuel_burn, d.conflicts, d.departure_delay, d.arrival_delay, d.distance, d.congested_sector 
from decision
where d.treatment=? and d.execution=? 
// Generalize the results by getting also info from other times 
and d.time>=? and d.time<=? 
and d.routes_seq=r.seq 
and r.origin=? 
and r.dest=?;";
// Allow information from other airlines also: no restriction for airline

private static final String ADAPT_SQL_LOCAL = "select d.fuel_burn, 0 as conflicts, d.departure_delay, d.arrival_delay, d.distance, 0 as congested_sector 
from decision
where d.treatment=? and d.execution=? 
// Generalize the results by getting also info from other times 
and d.time>=? and d.time<=? 
and d.routes_seq=r.seq 
and r.origin=? 
and r.dest=?;";

/**
 * Performs the epsilon-Q learning process on this airline
 * @param universe The @link FACETExperiment object , the data repository
 * @param systemInfoAvailable If true the airline has access to the system-wide information too.
 * If false the airline only has access to its own data.
 * @param connection The database connection to the database
 * @return the total number of adapted records
 */
private int adapt(FACETExperiment universe, boolean systemInfoAvailable, java.sql.Connection connection) {
    if (lambda <= 0) // If lambda == 0 then the value of the reward has no effect on the Q-functions (no learning!!)
        return 0;
}```
metrics.changedqRecs = 0;

if (choices.isEmpty()) {
    return 0;
} else {
    java.sql.PreparedStatement stmt = null;
    int adaptedFlights = 0;

    try {
        // A trick to make the system ignore the information about the the system metrics (even if it is in the database)
        if (systemInfoAvailable) {
            stmt = connection.prepareStatement(ADAPT_SQL_GLOBAL);
        } else {
            stmt = connection.prepareStatement(ADAPT_SQL_LOCAL);
            stmt.setString(7, _name);
        }

        for (Flight flight : flights) {
            // Generalize the time (remove year, month, and day info)
            short minutes = TimeMapper.getMinutesOfDay(flight.getSchDepTime());
            // Now find the corresponding element of the Q-function
            QFunctionRecord qRec = choices.get(QFunctionRecord.computeHash(minutes, flight.getRoute()));
            if (qRec != null) {
                double value = computeReward(universe, flight, stmt);
                double rValue = qRec.getValue();
                if (updateQFunction(qRec, value)) {
                    adaptedFlights++;
                    // Recover the new value and check if the change was significant
                    value = qRec.getValue();
                    if (rValue == 0 || (Math.abs((value - rValue) / rValue) > FACETExperiment.changeQThreshold)) {
                        metrics.changedqRecs++;
                    }
                }
            }
        }
    } catch (java.sql.SQLException e) {
        System.err.println("Error adapting for airline: "+_name + ";" + e.getLocalizedMessage());
    }

    finally {
        try {
            if (stmt != null)
                stmt.close();
        } catch (java.sql.SQLException e) {
        }
        return adaptedFlights;
    }
}

/**
 * Computes the rewards for a flight of this airline.
 * <p>The effects of data availability, accuracy, and latency are introduced
The database contains the real data, but the airline will 'see' the degradation in the information.

```java
private double computeReward(FACETExperiment universe, Flight flight, java.sql.PreparedStatement stmt) {
    try {
        // There is no actual route to reward
        if (flight.getRoute() == null)
            return 0;

        // Compose the sql statement
        short minutes = TimeMapper.getMinutesOfDay(flight.getSchDepTime());
        stmt.setShort(1, universe.getTreatment());
        stmt.setInt(2, universe.getExecution() - universe.getLatency()); // This implements the latency
        stmt.setShort(3, (short)(minutes - 15));
        stmt.setShort(4, (short)(minutes + 15));
        stmt.setString(5, flight.getRoute().getOrigin()); // util.AirportCodeConverter.from4to3
        stmt.setString(6, flight.getRoute().getDest()); // util.AirportCodeConverter.from4to3
        java.util.ArrayList<double[]>*his*fl*ig*h*metrics = new java.util.ArrayList<double[]>();
        java.util.ArrayList<double[]> flight_metrics = new double[]();
        while (result.next()) {
            // introducing negatives because I am minimizing (inverting scales to use the same domination operator)
            flight_metrics = new double[]{
                -Math.round(result.getDouble("fuel_burn") * (sigmaFactor * _random.nextGaussian() + 1)),
                -Math.round(result.getInt("conflicts") * (sigmaFactor * _random.nextGaussian() + 1)),
                -Math.round(result.getInt("departure_delay") * (sigmaFactor * _random.nextGaussian() + 1)),
                -Math.round(result.getInt("arrival_delay") * (sigmaFactor * _random.nextGaussian() + 1)),
                -Math.round(result.getDouble("distance") * (sigmaFactor * _random.nextGaussian() + 1))
            };
            // Congested is a percentage (multiply by 100 otherwise round will return 0 or 1 only)
```
```java
Math.round(100 * result.getDouble("congested_sect") * (sigmaFactor * random.nextDouble() + 1));

historic_flight_metrics.add(flight_metrics);
}
result.close();

if (historic_flight_metrics.isEmpty()) // No need to compare to an empty population
    return 0;
else {
    // Obtain the vector for the current performance of the flight
    double[] bens = new double[]{
        // Introducing negatives because I am minimizing (inverting scales to use the same
        // domination operator)
        -Math.round(flight.getFuelBurned() * (sigmaFactor * random.nextDouble() + 1)),
        -Math.round(flight.getConflicts() * (sigmaFactor * random.nextDouble() + 1)),
        -Math.round(flight.getDepartureDelay() * (sigmaFactor * random.nextDouble() + 1)),
        -Math.round(flight.getArrivalDelay() * (sigmaFactor * random.nextDouble() + 1)),
        -Math.round(100 * flight.getPercentCongestedSect() * (sigmaFactor * random.nextDouble() + 1)),
    };

    return comparator.compareToPopulation(bens, historic_flight_metrics);
}
}
```
if (aircraft.isFlying(facetACID)) {
  if (status == Flight.FLIGHT_STATE.Scheduled || status == Flight.FLIGHT_STATE.GroundDelayed)
    // Record only the transition
    setActualTakeoffTime(aircraft.getTakeOffTimeStamp(facetACID));
  status = Flight.FLIGHT_STATE.Flying;
}

double[] coords = aircraft.getCoordinates(facetACID);
setFlownDistance(getFlownDistance() + utils.getGCDistanceNM(getLastLatitude(),
  getLastLongitude(), coords[0], coords[1]));
setFuelBurned(aircraft.getFuelBurned(facetACID));
setCoords(coords);

// Count # congested sectors the flight traverses (this is a SWIM provided metric)
int curr_sect = aircraft.getCurrentSectorIndex(facetACID);
if (curr_sect >= 0) {
  if (curr_sect != last_sect) { // New sector, count everything
    last_sect = curr_sect;
    num_sect++;
    if (sector.getSectorLoad(curr_sect) >= sector.getCapacity(curr_sect) * SECTOR_CAPACITY_THRESHOLD) {
      num_congested_sect++;
      last_congested = true;
    } else
      last_congested = false;
  } else if ((last_congested & sector.getSectorLoad(curr_sect) >= sector.getCapacity(curr_sect) * SECTOR_CAPACITY_THRESHOLD) {
    // The sector became congested while flying through it, but I am not double counting it (if it was already congested, do not count again)
    num_congested_sect++;
    last_congested = true;
  }
}
else if (aircraft.isLanded(facetACID)) {
  if (status != Flight.FLIGHT_STATE.Landed)
    setActualLandingTime(universe.getSimTimeStamp());
  if (universe.flightRecordsOut != null)
    universe.flightRecordsOut.println(toStringSpecial());
  status = Flight.FLIGHT_STATE.Landed;
  if (stoppableObject != null) {
    stoppableObject.stop();
    stoppableObject = null; // Try to recover the memory of the stoppable (without loosing the agent)
  }
}
else if (status == Flight.FLIGHT_STATE.Scheduled) {
  if (record.sch_dep_time < universe.getSimTimeStamp())
    status = Flight.FLIGHT_STATE.GroundDelayed;
  setActualTakeoffTime(universe.getSimTimeStamp()); // To see the ground delay grow in time
}
else if (status == Flight.FLIGHT_STATE.GroundDelayed)
  setActualTakeoffTime(universe.getSimTimeStamp()); // To see the ground delay grow in time
} else {
    System.err.println("Not flying, not landed (increase landed limit in config file)\n\nt\n\nAssuming it is landed.");
    if (status != Flight.FLIGHT_STATE.Landed) {
        setActualLandingTime(universe.getSimTimeStamp());

        if (universe.flightRecordsOut != null) {
            universe.flightRecordsOut.println(toStringSpecial());
        }
        status = Flight.FLIGHT_STATE.Landed;
        if (stoppableObject != null) {
            stoppableObject.stop();
            stoppableObject = null; // Try to recover the memory of the stoppable (without loosing the agent)
        }
    }
}
Bibliography
Bibliography


217


Curriculum Vitae

Guillermo Caldern-Meza obtained his bachelors in Electronics from the Instituto Tecnológico de Costa Rica, his masters in Electronics from the University of Bolton (United Kingdom) and the University of Paderborn (Germany) where he was distinguished as the Best Foreign Student of the year 2000. He has over twelve years of experience as a software engineer, software developer, software project manager, and systems integrator. He has participated in the design, and development of data collection systems, automatic inspection systems, parsers and interpreters for computer programming languages, simulators, and digital circuits. He has published 8 papers, and briefed his work, in national and international conferences. He has been a member of IEEE for more than 10 years, and a member of the Phi Beta Delta honor society. His is currently a Research Assistant at the Center for Air Transportation Systems Research (CATSR), George Mason University, where he has been in contact with public and private institutions related to the aviation industry like NASA, Metron Aviation, Intelligent Automation, Sensis Corporation, FAA. In the CATSR, he is currently using simulators for the National Airspace System, Multi-Agent simulators, and Reinforcement Learning techniques to evaluate the effects of adaptive airline strategies in the presence of NextGen.