A learning-based approach for distributed planning and coordination of airport surface movement operations

Master of Science Thesis

Spyros Polydorou

Technische Universiteit Delft

TU Delft

A learning-based approach for distributed planning and coordination of airport surface movement operations

Master of Science Thesis

by

Spyros Polydorou

For obtaining the degree of

Master of Science
in Aerospace Engineering

at Delft University of Technology

To be defended publicly on 30/09/2021.

Student number: 4355903

Thesis committee:		
	Prof. Dr G.C.H.E de Croon	TU Delft, Chairman
	Dr. O.A. Sharpanskykh	TU Delft, Supervisor
	Dr. E. van Kampen	TU Delft, External Examiner

TU Delft

Acknowledgements

This thesis report marks the end of an incredible seven-year journey at the Delft University of Technology. A journey that made me the adult and the engineer that I am today. During these years, not only did I get to enrich my academic and professional background, but I also got to discover myself. I step out of university life as a self-aware and independent grown-up. A student, I will always nonetheless be.

There are some people that helped me reach this fortunate position and to whom I would like to express my gratitude. First and foremost, to my supervisor Dr Alexei Sharpanskykh, who introduced me to the field of Agent-based modelling and shared with me his academic and scientific expertise. Who saw potential in me and whose support and interest in my thesis was never-ending.

I would also like to express my gratitude to my close friends. To Kostis for spending countless hours hearing me talking about agents and machine learning and for all the philosophical discussions that helped me put ideas into perspective. My roommates for being always there for me and showing interest in what I do. Kosta and Anesti, I would like to thank you for all the fun times we had together. I am sure these will continue in the future. A special thanks to Filia, who has been in my side all this time, and despite the distance that separates the two of us, her presence and support were always here.

Finally, I would like to thank my family. My parents, Torkan and Harry, for providing me with all the necessary conveniences to study abroad and my sister Jasmin for her never-ending love and support. This work is dedicated to them.

Spyros Polydorou
Delft, September 2021

Contents

List of Figures vii

List of Tables ix

List of Abbreviations xi

List of Symbols xiii

Introduction xv

I Scientific Paper 1

II Literature Study previously graded under AE4020 29

1 Introduction 31

2 Airport Surface Movement Operations at Schiphol Airport 33
2.1 Schiphol Collaborative Decision Management . . . 33
2.1.1 The Milestone Approach . . . 33
2.1.2 Pre-departure Sequence . . . 35
2.1.3 Collaborative Management of Flight Updates . . . 35
2.2 Summary of surface movement operations at Schiphol . . . 35
2.3 Schiphol socio-technical system . . . 36
2.3.1 Agent types . . . 36
2.3.2 Interaction between agents . . . 38
2.4 Runway configurations . . . 39
2.4.1 Runway configurations at Schiphol . . . 39
2.4.2 Factors influencing runway configurations . . . 39
2.4.3 Planning and execution of reconfigurations . . . 41
2.5 Uncertainty factors in surface movement operations . . . 44

3 Previous Research in ATO 47
3.1 Decentralization in Air Transportation - Udluft (2017) . . . 47
3.2 Agent-Based Modelling of an Airports Ground Surface Movement Operation - Noortmans (2018) 48
3.3 Decentralized Control for Resilient Airport Surface Movement Operations - Fines (2019) . . . 51
3.4 Research gap . . . 53

4 Learning Mechanisms for Surface Movement Operations 55
4.1 Opportunities for Learning . . . 55
4.1.1 Modelling Incoming Traffic . . . 55
4.1.2 Predicting Future Traversal Times . . . 55
4.1.3 Highway Generation . . . 56
4.1.4 Speed Profile Assignment . . . 56
4.1.5 Conclusion. . . . 56
4.2 Anticipatory vehicle routing . . . 56
4.3 Approaches for taxi-time prediction . . . 59
4.3.1 Queuing models . . . 59
4.3.2 Reinforcement learning . . . 59
4.3.3 Multiple linear regression . . . 60
4.3.4 Fuzzy rule based systems . . . 61
4.3.5 Other machine learning methods . . . 63
4.4 Conclusion . . . 65

5 Multi-Agent Path Planning 67
5.1 Background on Multi-Agent Planning . . . 67
5.2 A* based approaches . . . 69
5.2.1 Cooperative A* search . . . 69
5.2.2 Standley's improvements . . . 69
5.2.3 Approximate and optimal anytime algorithms . . . 70
5.2.4 Other approaches . . . 70
5.3 Rule based approaches . . . 71
5.3.1 Push and Swap . . . 72
5.3.2 Tree-based agent swapping strategy . . . 72
5.3.3 Push and Swap variants . . . 72
5.4 Hybrid approaches . . . 72
5.4.1 Flow Annotation Replanning . . . 73
5.4.2 Multi-Agent Path Planning . . . 73
5.5 Reduction based approaches . . . 73
5.5.1 Constraint Satisfaction Problem . . . 74
5.5.2 SAT based solvers . . . 74
5.5.3 Integer Linear Programming . . . 75
5.5.4 Answer Set Programming . . . 76
5.6 Two-level based approaches . . . 76
5.6.1 M* family . . . 76
5.6.2 Increasing cost tree search . . . 77
5.6.3 CBS family . . . 77
5.7 Sampling based approaches . . . 82
5.8 Trade-off . . . 82

6 Research Proposal 85
6.1 Research Objective & Questions . . . 85
6.2 Research Scope . . . 88

III Supporting work 89

A Model Elaboration 91
A.1 Modelling assumptions . . . 91
A.2 Relevant agent properties . . . 91
A.3 Simulation Parameters . . . 93
A.4 Forward simulation algorithm . . . 94
A.5 Bayesian optimisation algorithm . . . 94
A.6 Input flight schedule details . . . 95
A.7 Input flight schedule details for conflict analysis . . . 96

B Simulation results 99
B.1 Taxi-time results for individual days . . . 99
B.2 Taxi-distance results for individual days . . . 102
B.3 Taxi-speed results for individual days . . . 104
B.4 Link participation per sampling strategy . . . 106
B.5 Absolute prediction error for Scenarios B and D . . . 110
B.6 Absolute error distributions for all mechanisms and scenarios . . . 111

Bibliography 115

List of Figures

2.1 A-CDM processes . . . 33
2.2 Recommended milestones as described in the Eurocontrol Airport CDM Implementation Manual. Retrieved from [30]. . . . 34
2.3 Collaborative Pre-Departure Sequence Planning. Retrieved from [71]. . . . 35
2.4 Ground operations flow in an airport. Retrieved from [47]. . . . 36
2.5 Diagram of the socio-technical system responsible for the ground operations at Schiphol airport. Retrieved from [60]. . . . 37
2.6 Runway availability windrose at Schiphol airport. Retrieved from [66]. . . . 40
2.7 Average wind speeds and direction present at Schiphol airport for the period between 08/2011 - 01/2020. Retrieved from [94]. . . . 40
2.8 Windrose of Schiphol airport. . . . 40
2.9 L_{den} average noise contour. . . . 43
2.10 L_{night} average contour. . . . 43
2.11 Expected average noise level for the flight year of 2019. Retrieved from [22]. . . . 43

3.1 Taxiway layout in the baseline model. Retrieved from [93]. . . . 48
3.2 Original layout . . . 50
3.3 Simplified layout . . . 50
3.4 The original layout on the left and the layout considered in the agent based model on the right. Retrieved from [60]. . . . 50

4.1 The link traversal time is calculated using the intention levels maintained by the infrastructure agent and congestion information from downstream agents. Retrieved from [17]. . . . 58
4.2 Concept of operations of the dMAS at an airport. Based on observed link traversal taxi times and intentions received by other ATC agents, agents maintain a ML model which maps intention levels to link traversal times. . . . 59
4.3 Structure of a typical FRBS. Retrieved from [46]. . . . 62
4.4 Comparison of the models' predictive perfomance on ARN and ZRH data. Retrieved from [65]. . . 63
4.5 Predicted taxi time error distributions for different traffic flows and weather conditions at CLT. Retrieved from [53]. . . . 64
4.6 An example of a macroscopic network topology model of a taxi process. Retrieved from [104]. . 65
4.7 The values of the machine learning predictors based on the macroscopic network topology model. Retrieved from [104]. . . . 65

5.1 Performance curves for the approximate algorithms. Retrieved from [83]. . . . 70
5.2 Performance curves for the optimal algorithms. Retrieved from [83]. . . . 70
5.3 An example of the priority decision making. Agents (1,2,3,4) with identified paths (a,b,c,d) have lengths (25,10,20,30) respectively. At each round agents update their knowledge of other agents' paths. . . . 71
5.4 Example of an alternative path Ω_i to a precomputed path $\pi(u)$ of an agent u. . . . 73
5.5 An example of a CPF problem and its expansion graph $Exp_\tau(G,4)$ consisting of 5 time layers. Retrieved from [88]. . . . 75
5.6 Success rate and number of solved instances on a 8x8 grid map with 10 agents. Retrieved from [89]. . . . 75
5.7 Number of solved as a function of runtime. Retrieved from [89]. . . . 75
5.8 The performance of uMDD-SAT and other suboptimal alternatives. Retrieved from [90]. . . . 76
5.9 ICT for three agents. The dashed lines represent duplicate child node which can be pruned. Retrieved from [72]. . . . 77

5.10 The success rate (y-axis) of CBS in the brc202d DAO map for increasing number of agents (x-axis). Retrieved from [74]. 78
5.11 The success rate (y-axis) of MA-CBS(B) using EPEA* as the low-level solver on the brc202d DAO map for increasing number of agents (x-axis). Retrieved from [75]. 78
5.12 The success rate (left figure) and runtime perfomance (right figure) of several optimal algorithms when tested on the brc202d DAO map. Retrieved from [9]. 79
5.13 Success rate comparison between GCBS variants and optimal CBS. Retrieved from [5]. 80
5.14 Tested on the brc202d DAO map. w = 1.01 81
5.15 32x32 grid with 20% obstacles. w = 1.1 81
5.16 Success rate of ECBS compared to bounded CBS versions (a) and other bounded suboptimal algorithms (b). Retrieved from[5]. 81

A.1 Arrivals per hour for the 8 days of real world flight schedule. 95
A.2 Departures per hour for the 8 days of real world flight schedule. 95
A.3 Arrivals per hour for 4th and 13th of May 2016. 96
A.4 Departures per hour for 4th and 13th of May 2016. 96
A.5 Number of movements (blue) and runway configuration occurrences (red) for 04-05-16. 97
A.6 Number of movements (blue) and runway configuration occurrences (red) for 13-05-16. 97

B.1 Taxi-time distributions per flight day between Baseline and dMAS CBS. 99
B.2 Taxi-time distributions per flight day between Baseline and dMAS-U(200) CBS. 100
B.3 Taxi-time distributions per flight day between Baseline and dMAS-U(300) CBS. 100
B.4 Taxi-time distributions per flight day between Baseline and dMAS-U(400) CBS. 101
B.5 Taxi-distance distributions per flight day between Baseline and dMAS CBS. 102
B.6 Taxi-distance distributions per flight day between Baseline and dMAS-U(200) CBS. 102
B.7 Taxi-distance distributions per flight day between Baseline and dMAS-U(300) CBS. 103
B.8 Taxi-distance distributions per flight day between Baseline and dMAS-U(400) CBS. 103
B.9 Taxi-speed distributions per flight day between Baseline and dMAS CBS. 104
B.10 Taxi-speed distributions per flight day between Baseline and dMAS-U(200) CBS. 104
B.11 Taxi-speed distributions per flight day between Baseline and dMAS-U(300) CBS. 105
B.12 Taxi-speed distributions per flight day between Baseline and dMAS-U(400) CBS. 105
B.13 Link participation when no undersampling was applied. 106
B.14 Link participation when undersampling ratio of 200 was applied. 107
B.15 Link participation when undersampling ratio of 300 was applied. 108
B.16 Link participation when undersampling ratio of 400 was applied. 109
B.17 Conflict prediction error distributions of Adaptive, Baseline and dMAS CBS in Scenario A. 111
B.18 Conflict prediction error distributions of dMAS-U(200), dMAS-U(300) and dMAS-U(400) CBS in Scenario A. 111
B.19 Conflict prediction error distributions of Adaptive, Baseline and dMAS CBS in Scenario B. 112
B.20 Conflict prediction error distributions of dMAS-U(200), dMAS-U(300) and dMAS-U(400) CBS in Scenario B. 112
B.21 Conflict prediction error distributions of Adaptive, Baseline and dMAS CBS in Scenario C. 113
B.22 Conflict prediction error distributions of dMAS-U(200), dMAS-U(300) and dMAS-U(400) CBS in Scenario C. 113
B.23 Conflict prediction error distributions of Adaptive, Baseline and dMAS CBS in Scenario D. 114
B.24 Conflict prediction error distributions of dMAS-U(200), dMAS-U(300) and dMAS-U(400) CBS in Scenario D. 114

List of Tables

2.1 Runway availability per traffic flow. 39
2.2 Commonly used runway configurations at Schiphol based on 1 month of operations during the winter 2018. The configurations are given in (X+Y) format where X are the arrival runways and Y the departing. In case of multiple arrival and/or departing runways a comma (,) is used to separate them. Retrieved from [36]. 39

3.1 Specification of aircraft kinematics. Retrieved from [60]. 49

4.1 The prediction accuracy of the RL model when trained and tested on different US airports. Retrieved from [4, 38]. 60
4.2 Summary of the models including the type of variables used. Retrieved from [50]. 61
4.3 Percentage of predicted taxi times within 1 minute of the actual taxi times for DFW data. Retrieved from [50]. 61
4.4 Comparison between the Mamdani FRBS and Linear regression on predictions made for ZRH. Retrieved [13]. 62
4.5 Taxi time predictions within ±5 minutes for departing aircraft in CLT. Retrieved from [53]. 64

5.1 Quantitative trade-off of multi-agent path planning algorithms. 83
5.2 Qualitative trade-off of multi-agent path planning algorithms. Red color is assigned to the unsuitable algorithms. Orange color is assigned to less suitable algorithms. Green color is assigned to the most suitable algorithms. 84

A.1 ATC related simulation parameters. 93
A.2 Aircraft agent related simulation parameters. 93
A.3 Other parameters used in the simulations. 93

B.1 Statistics of the absolute prediction error for prediction made in Scenario B. 110
B.2 Statistics of the absolute prediction error for prediction made in Scenario D. 110

List of Abbreviations

AO	Airport Operator
AAS	Amsterdam Airport Schiphol
ABM	Agent-Based Modelling
ARN	Stockholm-Arlanda Airport
ATA	Actual Time of Arrival
ATC	Air Traffic Control
ATM	Air Traffic Management
A-CDM	Airport Collaborative Decision Making
ADAM	Adaptive Moment Estimation
ADS-B	Automatic Dependent Surveillance-Broadcast
AIBT	Actual In-Block Time
ALDT	Actual Landing Time
ANN	Artificial Neural Network
ANSP	Air Navigation Service Provider
AOBT	Actual Off-Block Time
AQLI	Aircraft Queue Length Indices
ASRT	Actual Start-Up Request Time
ATCO	Air Traffic Controller
ATOT	Actual Take-Off Time
ACARS	Aircraft Communications and Reporting System
ATFCM	ATM Flow & Capacity System
A-SMGCS	Advanced Surface Movement Guidance and Control System
BO	Bayesian Optimisation
BDI	Belief Desire Intention
CB	Conflict Based
CV	Cross Validation
CBS	Conflict Based Search
CLT	Charlotte Douglas International Airport
CPF	Cooperative Path Finding
CTOT	Calculated Take-Off Time
CPDLC	Controller-Pilot Data Link Communications
CPDSP	Collaborative Pre-Departure Sequence Planning System
DR	Dead Reckoning
DFW	Dallas/Fort Worth International Airport
DNN	Deep Neural Network
dMAS	Delegated Multi-Agent System
EFS	Electronic Flight Strips
ETA	Estimated Time of Arrival
ETD	Estimated Time of Departure
EHAM	ICAO code of Amsterdam Schiphol Airport
EIBT	Estimated In-Block Time
ELDT	Estimated Landing Time
EXOT	Estimated Taxi-Out Time
EUROCONTROL	European Organisation for the Safety of Air Navigation
FP	False Positive
FRBS	Fuzzy Rule-Based System
ICAO	International Civil Aviation Organisation
IFR	Instrument Flight Rules
KLM	KLM Royal Dutch Airlines
kNN	k-Nearest Neighbours
KPI	Key Performance Indicator
LR	Linear Regression
LVNL	Luchtverkeersleiding Nederland
ML	Machine Learning
MAE	Mean Absolute Error
MAS	Multi-Agent System
MGH	Main Ground Handler
MAPF	Multi-Agent Path Finding
MAPP	Multi-Agent Path Planning
MRPP	Multi-Robot Path Planning
NM	Network Manager
NN	Neural Network
NMOC	Network Manager Operations Centre
OD	Origin Destination
PF	Path Finding
PM	Point Merge
QNH	Barometric Pressure Level
RF	Random Forest
RL	Reinforcement Learning
RMO	Runway Mode of Operations
RWY	Runway
SID	Standard Instrument Departure
SVM	Support Vector Machine
SVR	Support Vector Regression
SCFI	Surface Cumulative Flow Indices
SIFI	Surface Instantaneous Flow Indices
SRDI	Slot Resource Demand Indices
SESAR	Single European Sky ATM Research
TP	True Positive
TOBT	Target-Off Block Time
TSAT	Target Start-Up Approval Time
TTOT	Target Take-Off Time
TSK FRBS	Takagi and Sugenos FRBS
UBR	Utility Based Regression
VS	Visual Separation
VHF	Very High Frequency
WTC	Weight Turbulence Category
ZRH	Zurich Airport

List of Symbols

Symbol	Definition	Units & Constants
α_n	Acquisition function	[-]
$\hat{\beta}$	Estimated regression coefficient	[-]
δ	Time difference between link timeline samples	[-]
ϵ	Suboptimality factor	[-]
η	Learning rate	[-]
η	Makespan bound	[-]
λ	Mean arrival rate	[-]
$\mu_{A_i^l}(x_l)$	Membership function	[-]
$\phi(y)$	Relevance function	[-]
ϕ	Activation function	[-]
$\phi(\mathbf{Y})$	Sigmoid-based activation function	[-]
a_i	ith arrival	[-]
d_i	ith departure	[-]
int_{freq}	Intention frequency	$[s]$
t	Simulation timepoint	$[s]$
t_0	Estimated time of arrival of other aircraft	$[s]$
t_s	Estimated time of arrival for the commanded aircraft	$[s]$
t_R	Relevance threshold	[-]
v_{max}	Maximum taxi speed	$[\frac{m}{s}]$
v_{turn}	Maximum turning speed	$[\frac{m}{s}]$
w	Suboptimality factor	[-]
x_i	ith independent variable	[-]
y_i	ith dependent variable	[-]
$\hat{y}$	Predicted value	[-]
A^*	A star algorithm	[-]
A_i^l	Linguistic sets	[-]
AC_c	Commanded aircraft a conflict pair	[-]
AC_o	Other aircraft in a conflict pair	[-]
AC_{tt}	Aircraft average taxi time	$[min]$
AC_{td}	Aircraft average taxi distance	$[km]$
AC_{ts}	Aircraft average taxi speed	$[\frac{m}{s}]$
C^*	Optimal solution	[-]
F	Flight schedule	[-]
L_{den}	Dayeveningnight noise level	dB
L_{night}	Night noise level	dB
N	Number of aircraft	[-]
N^P	Number of aircraft pushed back before	[-]
N_P	Number of aircraft pushed back after	[-]
N_u	Number of hidden neurons	[-]
Q	Queue size	[-]
R^2_{adj}	Adjusted coefficient of determination	[-]
R_i	ith rule in the fuzzy-rule based system rule base	[-]
T_{req}	Require time needed to avoid conflict	$[s]$
T_{window}	Conflict time window	$[s]$
$U^p_\phi(\hat{y}, y)$	Utility of prediction	[-]
V	Velocity	$[\frac{m}{s}]$

Introduction

This thesis falls in the domain of airport surface movement operations. This domain has extensively been studied over the years in the Air Transport and Operations department at Delft University of Technology. The objective is to formulate distributed control architectures for efficient and safe traffic management on airport surfaces in view of the increasing trends of passenger traffic and the associated bottlenecks that the current operations face.

This work is the first in the research line to use Machine Learning in such a control architecture. The proposed model combines an already implemented Multi-Agent Path Finding algorithm with a coordination pattern called a Delegated Multi-Agent System. The latter enables the handling of communication complexity in domains that require repetitive interactions between agents.

This thesis report consists of two parts: In Part I, the scientific paper is presented. Part II contains the Literature Study. Finally, Part III further elaborates the model and presents additional results that support the paper's findings.

I

Scientific Paper

A Learning-Based Approach for Distributed Planning and Coordination of Airport Surface Movement Operations

S. Polydorou*,
Dr. O.A. Sharpanskykh†

Delft University of Technology, Delft, The Netherlands

Abstract

Agent-based distributed planning and coordination has shown promising results in controlling operations in complex systems such as those present at airports. Distributed planning differs from centralised approaches because it is performed by several agents, which coordinate plans with each other in order to meet a global objective. In this research, we examine airport surface movement operations and focus specifically on improving the conflict detection abilities of a Multi-Agent Path Finding (MAPF) approach for distributed planning and coordination using machine learning. Our MAPF proposal is built on top of a distributed CBS-based algorithm implemented in an existing Multi-Agent System (MAS) model of Amsterdam Schiphol Airport (AAS). In the proposed approach, we use a delegated Multi-Agent System (dMAS), firstly, to propagate information related to the intended aircraft paths and, secondly, to perform the conflict detection task of the CBS algorithm. To achieve these, the dMAS accesses a set of Artificial Neural Networks (ANNs), each allocated to specific taxiway segments to obtain traversal time estimates of aircraft intending to use those segments. Propagated aircraft intentions are used as predictors for future traversal time predictions either during the intention propagation phase or during CBS conflict detection. The proposed planning and coordination model and three of its variants were tested on a real-world flight schedule extracted from ADS-B ground tracks. Comparisons with a baseline approach that implements a forward kinematic simulation for conflict detection revealed that dMAS-CBS offered more precise conflict predictions while being less prone to Type I errors. More specifically, under scenarios where the airport operates at peak capacity, dMAS-CBS was twice as precise and produced up to five times fewer false positives predictions than the baseline approach.

1 Introduction

Despite the impact that COVID-19 has had on the aviation industry, reflected in the significant reduction in traffic levels, the number of Instrument Flight Rules (IFR) flights is expected to rise in the coming years. The current estimate is that by 2024 the traffic levels of 2019 will be reached [1], and from then on, increasing trends are expected. A consequence of these increasing trends is the appearance of bottlenecks in the Air Traffic Management (ATM) system. These are mainly driven by the limited capacity and resources available to accommodate all the traffic. One of the critical areas are airports. In the third quarter of 2019, 9% of the total delay encountered in the European network was caused by delays originating from airport operations [2]. Therefore, there is a strong need to invest in new technologies and find new solutions to overcome these imminent issues.

Currently, airport movement operations are handled by Air Traffic Controllers (ATCOs) in a way that resemble operations of a centralised system. Furthermore, humans are limited in terms of their mental capacity, and hence their workload and performance can vary depending on the conditions existing on the airport's surface. An area of research aiming at enhancing the performance of ATCOs is the design of autonomous systems with the ability to perform some of the ATCOs' tasks while meeting operational targets such as minimising the taxiing time and distance. This, however, is a challenging task since factors such as size and airport layout, number of aircraft taxiing and weather conditions influence the complexity of the problem. In addition, one needs to capture the various underlying dependencies between taxiing aircraft.

An approach that has shown promising results with respect to increasing the operational efficiency of complex systems is agent-based distributed planning and coordination. In the context of airport surface movement operations, decentralisation has shown to be effective at managing movements of aircraft on the airport's

*Msc Student, Air Transport and Operations, Faculty of Aerospace Engineering, Delft University of Technology
†Assistant Professor, Air Transport and Operations, Faculty of Aerospace Engineering, Delft University of Technology

surface [3]. In [3, 4], decentralised bidding mechanisms were used to coordinate actions between virtual ATCOs. The proposed mechanisms could safely and efficiently handle chaotic operations, with only limited information exchange needed between the agents. In a follow-up study [5], a distributed version of the Conflict Based Search (CBS) [6] Multi-Agent Path Finding algorithm was implemented. CBS is a two-level algorithm that firstly determines and then resolves anticipated conflicts between the agents' plans. This is achieved by predicting the arrival time of the agents to segments of their route and imposing constraints if conflicts are found. Comparisons with real-world historical data showed that the algorithm improved the average aircraft taxi time and positively influenced the system's overall resilience to adverse events, such as runway reconfigurations.

In our study, we continue the work presented in [5], focusing specifically on the conflict detection performance of the CBS-based algorithm. More precisely, at each conflict detection cycle, the algorithm performs a forward simulation of an aircraft's path, predicting its taxi time to various locations of the taxiway system. The simulated paths are then compared with simulated paths of other aircraft to identify possible conflict points. The predictions made during the simulations consider the aircraft dynamics when taxing in straight and turn segments. The approach demonstrated good behaviour and solved all of the predicted conflicts for a given flight schedule. However, certain deficiencies in the conflict prediction method are thought to impact conflict detection precision negatively. Firstly, the algorithm does not consider the underlying dependencies between aircraft when calculating their arrival time to a specific location. In addition, the effect of the air traffic controller agent (ATC Agents) actions when controlling an aircraft is also not taken into account. These issues can ultimately shift the predicted conflict detection timepoint leading to false positives and negative predictions. A suitable candidate to overcome limitations related to prediction imprecision is Machine Learning (ML). A learning model able to capture the underlying relationships between aircraft and the consequences of particular agent actions is hypothesised to enhance the effectiveness of a distributed airport control system. For this reason, the objective of this research is formulated as follows:

> *To study the ability of a cooperative multi-agent path planning algorithm for distributed airport surface movement operations by improving its conflict detection performance using a machine learning mechanism.*

Within the domain of airport surface movement operations, most studies implementing ML methods have focused on taxi-time prediction. The proposed techniques include queuing models [7], reinforcement learning [8], multiple-linear regression models [9], fuzzy rule-based systems (FRBS) [10], and other popular algorithms such as Support Vector Machines, k-Nearest Neighbours (kNN), Random Forests (RF) and Neural Networks (NN) [11]. According to the authors' knowledge, there has not been any research that has studied the effects of learning in controlling airport ground movements in a distributed coordination context. One study with a similar objective and MAS set-up has been performed in the domain of road traffic control. More specifically, in [12] the use of an adaptive anticipatory vehicle routing system to control road traffic was proposed. This MAS-based approach aims to provide users of a road network with traffic information about the most optimal routes to take. The idea was realised using the technique of delegated Multi-Agent Systems (dMAS) [13, 14]. dMAS is a coordination technique that alleviates the complexity of direct communication protocols between agents. Instead, the complexity is delegated to a dedicated behaviour module called Mobile Agent, which exhibits agent-like characteristics.

In this study, the MAS model specifications of the Amsterdam Airport Schiphol in [5] have been expanded to incorporate the principles of dMAS. Our proposed coordination and planning approach combines dMAS with a distributed version of the CBS MAPF algorithm. For the remainder of the paper it will be referred to as dMAS-CBS. More specifically, the Mobile Agents within dMAS are delegated to inform downstream ATC Agents about the estimated arrival time of aircraft to taxiway segments under their control. This constitutes the intention propagation phase. The Mobile Agents calculate the estimated arrival time by performing queries on a set of single-layered artificial neural networks (ANNs). Each ANN is responsible for modelling the traversal time dynamics of specific taxiway segments while their predictions rely on the count of previously propagated intentions. All ANN models were trained on data generated by running dMAS on semi-random flight schedules. The ability of the Mobile Agents to traverse the environment and perform sequential queries is then used in the CBS algorithm to perform conflict detection.

Moreover, due to the sporadic occurrence of conflicts in the taxiway environment, making them less represented in the data, we study the possibility of applying undersampling to bias the algorithms' learning process. With undersampling, the amount of less relevant samples in the datasets is decreased so that the learning algorithm gives a higher emphasis on the more relevant samples. In our context, we define as relevant a data sample that corresponds to a longer link traversal time than average.

All dMAS variants were compared with the baseline approach using a real-world flight schedule. The MAS simulations showed that dMAS-CBS was able to control aircraft traffic effectively, but its effect on the overall system behaviour compared to the original approach was insignificant. When looking at the conflict detection performance, it is found that, on average, the prediction error of dMAS-CBS is smaller than the

baseline approach, and it is less susceptible to Type I errors. More specifically, under scenarios in which the airport operates at peak capacity, the dMAS-CBS mechanism is significantly more precise at predicting conflicts. Regarding the effectiveness of the undersampling variants, the results indicate that reducing the training data size negatively influences the learners' behaviour, leading to situations where they cannot safely control the ground traffic.

The remainder of this paper is structured as follows. Section 2 provides a global overview of the proposed model. After that, the specifications of the model are presented in Section 3. Details regarding ANN properties and training are presented in Section 4, while the ways by which the MAS model verification and validation were incorporated in the study are discussed in Section 5. The results and analysis of the MAS simulations are presented in Section 6 followed by a discussion and recommendations in Section 7. Finally, the conclusions of this research are presented in Section 8.

2 Global overview of model

This research is based on a MAS model of AAS, developed in previous studies [3, 4, 5]. Our model consists of a Multi-Agent Path Finding algorithm enhanced by delegated Multi-Agent Systems and Machine Learning. This section presents the two components and provides an overview of the proposed agent interactions.

2.1 Multi-Agent Path Finding

The original MAS model [1] implements a distributed planning and coordination mechanism that operates similarly to the CBS MAPF [6] algorithm. The mechanism's goal is to find the shortest conflict-free path for each aircraft while considering the cost of all paths and the availability of taxiway segments. The shortest path calculation is performed using the A* algorithm [15]. At the same time, aircraft conflicts are handled in a two-step manner, specifically in the conflict detection and resolution phase. The two phases are incorporated into the CBS and ATC agents' behaviour, respectively. Our research focuses on improving conflict detection, which is currently done analytically through a forward kinematic simulation algorithm. The algorithm calculates the estimated time of arrival (ETA) of all aircraft to every ATC Agent along their route by propagating their instantaneous dynamics and adjusting them in case of turn segments. The algorithm can be found in Appendix A of [16].

2.2 Delegated Multi-Agent System

Delegated MAS [13, 14] is a type of distributed coordination mechanism inspired by ant behaviour and Belief Desire Intention (BDI) agents. dMAS has shown to be able to achieve complex coordinated behaviour in coordination-and-control applications such as manufacturing control and planning of manufacturing plants [14], allocation and composition of services in cloud environments [17, 18] and in anticipatory vehicle rooting [12, 19]. These are all domains that require repetitive interactions between Task Agents and Resource Agents. The former represent tasks that need to be accomplished, while the latter represent the various resources necessary for achieving these tasks. Such repetitive interactions can create communication bottlenecks, especially in large-scale applications [12]. This complexity can be handled by delegating these interactions to a separate MAS, called dMAS. A dMAS relies on the concept of an intelligent message or Mobile Agent. In essence, it is a self-contained entity with agent-like characteristics, able to carry information and move autonomously through a virtual environment. Our proposed methodology is inspired by the use of dMASs in the domain of anticipatory vehicle rooting [12].

An overview of the proposed concept and the relevant agents and interactions is shown in Figure 1. A dMAS is used in two ways; to propagate intentions and to detect conflicts. During the intention propagation phase, the ATC Agents use Mobile Agents to inform other ATC Agents about the ETA of aircraft under their control. The same intentions are then used as predictors for future link traversal times queries. In parallel, the CBS Agent uses the Mobile Agent to retrieve the ETA estimates allowing it to perform the necessary conflict detection tasks. In both cases, ML is used to estimate the traversal times of the Aircraft Agents to future locations on the taxiway system.

3 MAS model specifications

In this section, the specifications of the learning-based model are described. The model consists of an abstraction of the taxiway system at AAS and several types of agents. First the specifications of the environment are provided followed by the specifications of the agents' local properties and interactions.

[1] We refer to as original the MAS model presented in [5]

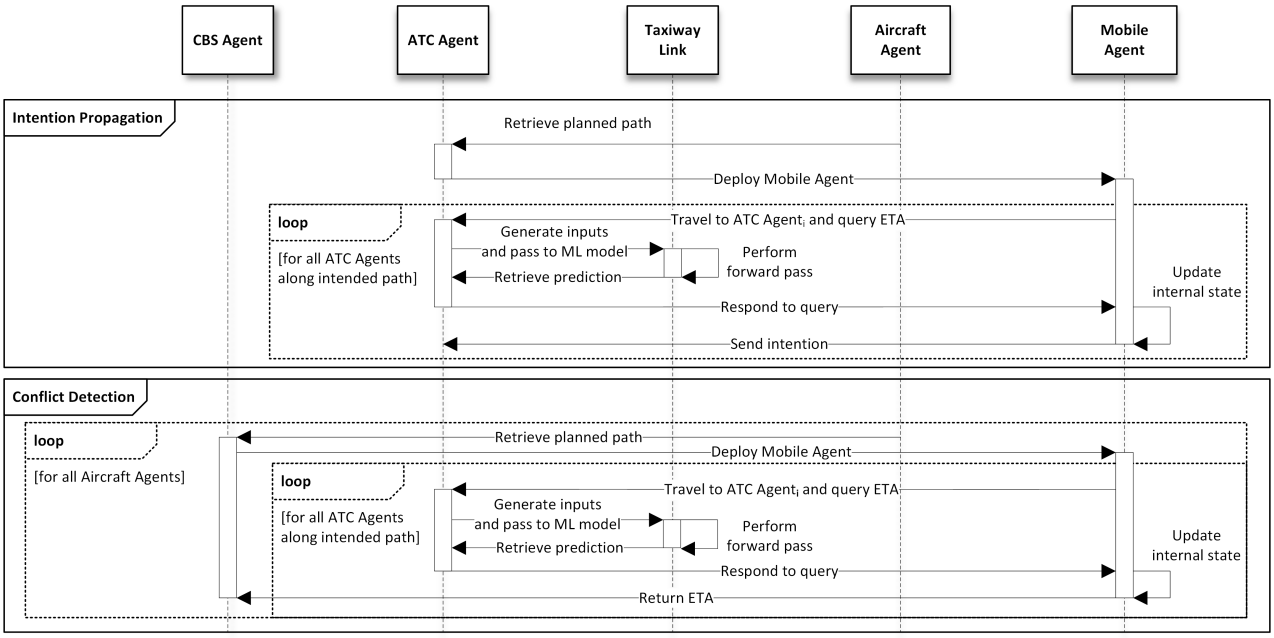

Figure 1: Sequence diagram showing the agent interactions during the phase of intention propagation and conflict detection.

3.1 Environment Specifications

The MAS environment consists of a graph, $G(V, E)$, representing the taxiway and runway layout of AAS. The graph model, shown in Figure 2, consists of graph nodes placed in taxiway intersections, pier and runway entry/exit points. The nodes are connected with graph edges and together make up the taxiway system. The graph edges can either be bi-directional or uni-directional and can be added or removed. Moreover, each edge has a weight associated with it which is used by the A* algorithm for new route calculation purposes. Upon graph initialisation, all edges are made bi-directional, and weights are assigned according to the estimated time it takes for an aircraft to taxi along the edge at its maximum taxi speed. Further details regarding the choices behind weight assignment can be found in [4]. The directionality, removal/addition and weight of each edge are controlled by ATC agents situated on the nodes of the graph model.

Additionally, a flight schedule F is used as a static environment object that can be accessed at any time point by the Entry/Exit Agents and the Airport Operation Status Agent. Each row of F contains the flight's ID, entry and exit point in the airport layout, and the scheduled release time to the taxiway system.

To incorporate dMAS into the model, the environment has been designed to include a set of links each representing a taxiway segment. We distinguish between two types of links, namely $Link$ and $LinkModel$.

Links of $LinkModel$ type are dynamic environment objects, meaning that their state can change during the simulation. This occurs based on the information propagated by the Mobile Agents. $LinkModel$ links hold an intention database which can be accessed at any time point by the ATC Agents located at their nodes. The content of this database is filtered every time point to remove possible outdated and duplicate intentions. Moreover, links of $LinkModel$ type hold an ML model as an attribute. When queried by a Mobile Agent, the Link constructs a vector of features and feeds it to its ML model. The ML model then returns a prediction of traversal time. Included in these features are past intention levels. These are generated by processing the Link's intention database. The process involves creating a timeline of intention level counts, based on the database contents, re-sampled in 30s intervals. Depending on a user-defined input, the intention count from the respective intervals is then sampled. A default value of 6 has been used for our research. The time-difference δ, between subsequent samples, is set equal to the interval length. Figure 3 shows how this concept works. Links of $Link$ type also return responses to traversal time queries. Unlike $LinkModel$ links, they do not have any prediction logic and always return the average observed aircraft traversal time. Links representing runway exits and gate segments are of this type. The reason is that aircraft on runway exits have priority over traffic on neighbouring links, and aircraft are released from their gates only if traffic is not present on neighbouring links [5]. Hence, the traversal time uncertainty for such links is expected to be low, and assigning ML models is unnecessary.

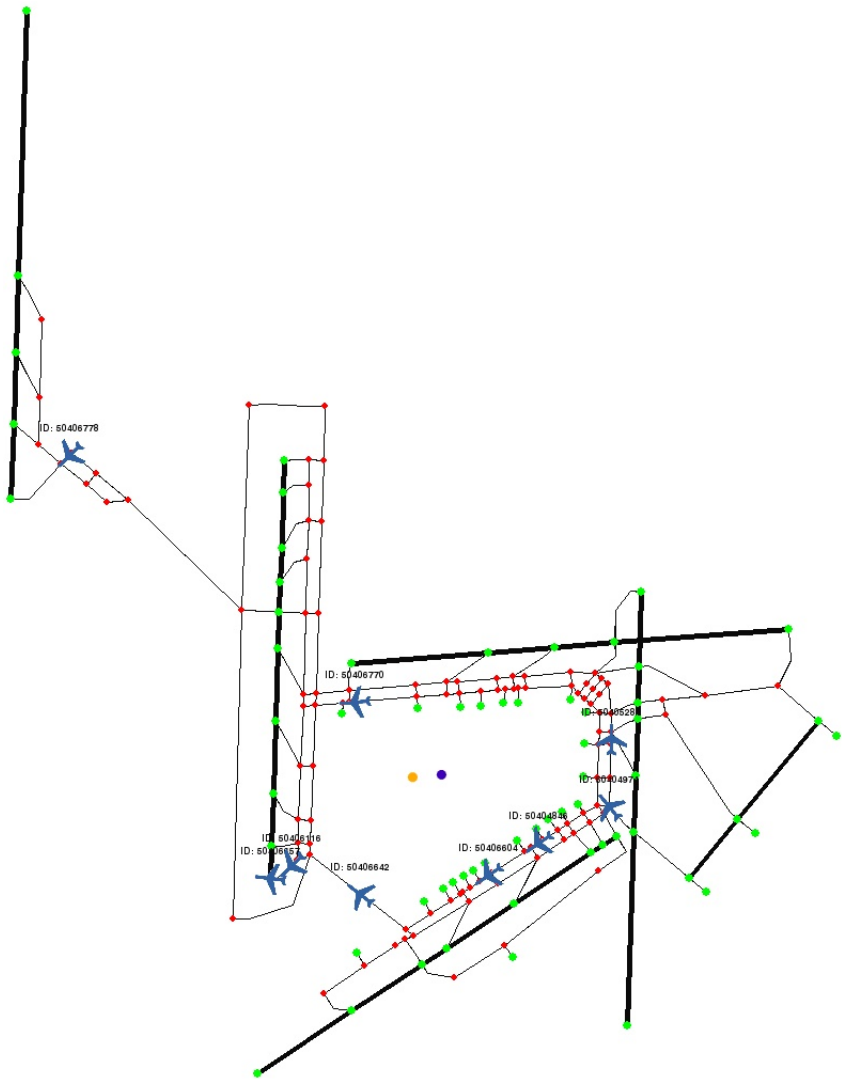

Figure 2: Environment of the AAS agent-based model. The red dots represent the location of the ATC Agents. Exit/Entry Agents are located on the green dots. The yellow dot represents the location of the CBS Agent, and the purple dot the location of the Airport Operation Status Agent. The blue aircraft are the Aircraft Agents.

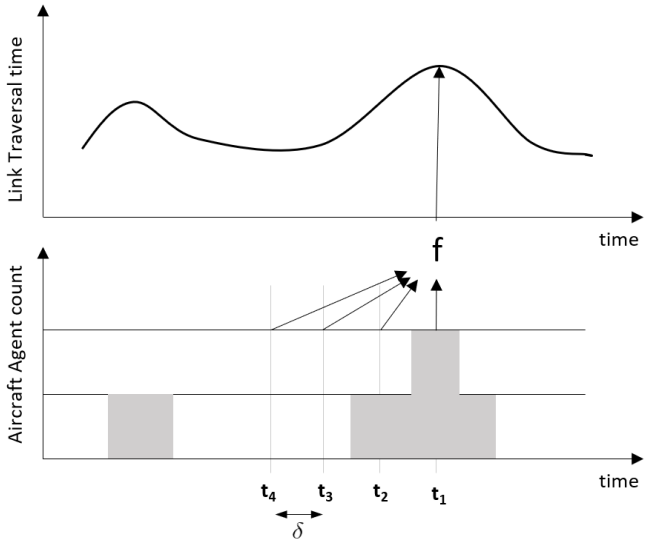

Figure 3: Diagram depicting the concept of using past intention level as predictors of a function that represents the link traversal time over time.

3.2 Agent Specifications

3.2.1 Entry/Exit Agents

Entry and exit agents are responsible for safely inserting and removing aircraft agents from the simulation environment. They are located at the pier entry/exits and runway holding points, and both have access to the flight schedule. Entry agents aim to release aircraft agents close to their planned spawn time, as specified in the flight schedule, and are responsible for computing an initial path for them using the A* algorithm. After an Aircraft Agent has been handled, the entry or exit agent imposes an occupancy time on the runway or gate, similar to real-world operational practices.

3.2.2 Aircraft Agents

Aircraft Agents are responsible for the timely execution of the ATC agents' commands. An Aircraft Agent's primary goal is to taxi along its assigned route and reach its destination as soon as possible. While doing so, it needs to adhere to its operational limits. The Aircraft Agents' dynamical properties, such as maximum/comfort acceleration/deceleration and taxi speed, are based on the A320/B737 type of aircraft. The behaviour of the Aircraft Agent can be summarised in two main properties. These are:

Execute ATC commands: This property involves an interaction between the Aircraft Agent and the ATC Agents. ATC Agents communicate speed or route related commands to the Aircraft Agents. When instructing a speed command, the ATC Agent specifies the distance within which the command is required to be satisfied. When no command is given, the Aircraft Agents accelerate to their maximum speed v_{max}. Route commands usually involve a change in heading. In case the required heading change is more than 30°, the Aircraft Agent adjusts its speed to v_{turn}.

Maintain Separation: This property involves an interaction between an Aircraft Agent and other Aircraft Agents. Aircraft Agents are equipped with a radar that enables them to observe their surroundings, up to 250 meters, and maintain separation from other aircraft. The reasoning behind this is to resemble the pilot's ability to see his/her surroundings, enabling visual separation to be kept if needed. If separation is lost, the agent's internal logic determines the type of conflict, either of the following type or crossing. In a situation where the two agents are following each other, the agent responsible for maintaining separation is the one behind. In the case of a crossing conflict, the agent nearest to the crossing node is given priority. Subsequently, the property determines an appropriate deceleration level to maintain the minimum separation between the aircraft.

3.2.3 CBS agent

The CBS agent is responsible for executing the planning mechanism's conflict detection tasks. At each simulation time point, the CBS agent gathers all the intended paths of the Aircraft Agents taxiing on the graph and predicts the time travel required by them to reach each of the nodes along their path. In the event a pair of Aircraft Agents is anticipated to pass by an ATC Agent, within a user-defined conflict time window (T_{window}), irrespective of their direction of travel, a conflict is declared at the ATC agent's location. The CBS Agents then contacts the ATC Agent, who then updates his state with the newly acquired information.

To enable dMAS-based conflict detection, the conflict detection property of the original CBS agent specification is adapted. More specifically, the CBS agent will dispatch Mobile Agents for each conflict detection cycle in which dMAS is activated. The delegated agents will return the ETA of an aircraft to each ATC Agent that it intends to pass by and use it to compare with all other ETA combinations of other Aircraft Agents.

Upon initial trials of this concept, however, it was determined that the additional layer of complexity significantly increases runtime, especially when the density on the taxiway system is high. For each time point, an ML query for each link of all aircraft paths has to be made. In order to avoid incurring runtime issues, it was decided to limit the duration in which dMAS-CBS is used. To further support this action, we argue that the effect of ML is more appropriate to measure when the system undergoes a change of state due to either the addition of a new Aircraft Agent or as a result of a conflict resolution command. The dMAS-CBS is therefore activated for the time-points after such occurrences. The length of the interval that dMAS-CBS is activated was set to a default value of 20 seconds.

3.2.4 ATC Agents

The ATC Agents are responsible for safely and efficiently guiding the Aircraft Agents along their routes. Their collective behaviour realises a complete traffic guidance and surveillance system, using distributed coordination and planning principles. ATC agents take control of the aircraft agents approaching their location and command them accordingly. Upon termination of control, the controlling ATC Agent hands the flight over to the following ATC Agent along the aircraft's route. ATC Agents are placed on taxiway intersections and next to Entry/Exit Agents. The behaviour of the Aircraft Agent can be summarised in the following properties:

Determine Conflict Type: This property is executed whenever an ATC Agent receives information from the CBS agent about a potential conflict at its location. The agent will first determine the type of conflict and afterwards proceed with the appropriate resolution strategy. Initially, the agent determines whether the aircraft are heading towards each other from opposite directions. Such a conflict would create a severe gridlock, blocking traffic at a taxiway segment completely. To avoid such a situation, the agent executes the *Head-On Conflict Resolution Property*. If the agent determines that two Aircraft Agents are expected to cross nodes but not in a head-to-head fashion, it executes the *Crossing Conflict Resolution* property.

Head-On Conflict Resolution: This property defines an interaction between ATC and Aircraft Agents. The ATC Agent determines which Aircraft Agent is going to receive a re-route command. The choice is to send the command to the agent further away from the conflict location since, in general, there would be more alternative paths available for that agent. By removing the segment in which the conflict is anticipated to occur, the responsible ATC Agent determines a new route and communicates to the Aircraft Agent. The ATC Agent in command of that Aircraft Agent is also informed of the change and, depending on the newly determined path, communicates an appropriate heading command.

Crossing Conflict Resolution Property: This property defines an interaction between ATC and Aircraft Agents. The ATC agent executes this property in case of an anticipated crossing conflict. The agent determines which Aircraft Agent is the furthest away from the conflict location and sends a speed control command. The required speed V_{req} is calculated as follows:

$$T_{req} = t_s + T_{window} - (t_s - t_0) - t \qquad (1)$$

$$V_{req} = \frac{d_s}{T_{req}} \qquad (2)$$

where t_s and d_s are the Estimated Time of Arrival (ETA) and the distance to the conflict location respectively, for the Aircraft Agent that will receive the speed command. T_{window} is a user defined time window within which an anticipated conflict is declared, t is the simulation time point and t_0 is the ETA of the other aircraft to the conflict location.

Dispatch Mobile Agent: This property involves interactions between the Mobile Agent and ATC Agents. At user-defined time-points (int_{freq}), the agent dispatches a Mobile Agent for each Aircraft Agent under its responsibility. This guarantees that Aircraft Agents that spawn in the taxiway system are taken into account when the state of the ATC Agents is being updated.

Distribute Intentions: This property involves interactions between the ATC Agent and the environment. At every time point, the ATC agent determines whether it has received an intention message from a Mobile Agent as part of a dMAS issued by another ATC Agent. If the agent has received a message, it is read and then stored in the intention database of the link that the intention message is referring to. The ATC Agent subsequently discards the message.

Update Link Traversal Time: This property defines an interaction between the ATC Agent and the environment. For each link traversal, the ATC Agent located at the end of the link updates the state of that link with the traversal time of the Aircraft Agent. The measurement is used by links of *Link* type to update the average traversal time value which they will use to respond to future Mobile Agent queries.

3.2.5 Mobile Agent

Contact ATC Agents: This property defines interactions between:

- Mobile Agent and ATC Agents,
- Mobile Agent and CBS Agent

Mobile Agents are responsible for propagating the Aircraft Agent's intentions and querying link traversal time information from relevant links. An intention message contains the location of the Aircraft Agent and the estimated time of arrival (ETA) to that location, as well as an expiry time. The expiry time ensures that only valid intentions are stored in the links' intention databases. Intentions have to be sent in scenarios where the plan of the Aircraft Agent changes, for example, when a conflict resolution command has been sent or when the Aircraft Agent is performing a visual separation manoeuvre. ATC Agents use the received information to update the link intention databases.

The ATC Agent responsible for the control of the Aircraft Agent delegates this task to a Mobile Agent. The Mobile Agent queries traversal time estimates from each link along the way and, based on the response, notifies the ATC Agent that the Aircraft Agent will occupy that link between ETA and ETD. Through this intention propagation process, all ATC Agents of the taxiway system hold information regarding the aircraft's future location and time.

Mobile Agents are also instantiated when CBS is switched into dMAS mode of operation. During this period, instead of using the original forward simulation to compare aircraft paths, the task of determining the traversal time is delegated to a Mobile Agent. No transfer of the information regarding Aircraft Agents' intentions takes

place in this step. The Mobile Agent constructs a timeline of ATC Agent locations and corresponding ETAs, which then passes to the CBS Agent. The pseudo-code of the Mobile Agent's property is shown in Algorithm 1.

Algorithm 1 Contact ATC Agents

$A/C_{route} \leftarrow$ Aircraft Agent route
$A/C_{mode} \leftarrow$ Aircraft Agent mode of operations
$ATC_{app} \leftarrow$ ATC Agent which Aircraft Agent is approaching
$link_c \leftarrow$ current Aircraft Agent link
$X_c \leftarrow$ feature vector for current link traversal
$t \leftarrow$ timepoint Mobile Agent is generated
$ETA_{all} \leftarrow$ empty list

$ETA \leftarrow t + query(link_c, X_c)$ *// estimated time of arrival to approaching ATC Agent*
if A/C_{mode} **is** *arrival* **then**
  $destination_{a/c} \leftarrow A/C_{route} - 2$ *// excluding gate links*
else
  $destination_{a/c} \leftarrow A/C_{route} - 3$ *// excluding runway exit links*
end if
for each *link* **in** A/C_{route} **until** $destination_{a/c}$ **do**
  $ETD \leftarrow query(link, ETA, X)$
  if not CBS **then**
    $send\_intention(ETA, ETD)$ *// sending intention to ATC Agent controlling the link*
  end if
  $ETA \leftarrow ETA + ETD$ *// ETA to the next ATC Agent in the route*
  append ETA **in** ETA_{all}
end for
if CBS **then**
  return A/C_{route}, ETA_{all} *// return ETA to each ATC Agent along the route*
end if

3.2.6 Airport Operation Status Agent

This agent is responsible for determining the runway status at any given time point and communicating it to the ATC Agents. The ATC Agents can then add or remove certain edges, depending on the runway use so that runway incursions are avoided.

3.3 Changes in Original Planning & Coordination

By design, the original CBS implementation [5] was set to perform its conflict detection and resolution cycle at every time point. In this way, it adapted its conflict prediction estimates according to the evolving dynamics in the environment. The adaptive nature of the algorithm allowed for the evaluation of a particular conflict pair multiple times within a given period. According to [5], the algorithm could resolve all conflicts when given a real-flight schedule. Despite the suitability of the algorithm for separating traffic, a few concerns have been identified during our study. Firstly, the coupling of the proposed learning mechanism and the distributed CBS variant was found to increase the overall computational complexity, making the simulation challenging to run with the current computational resources. The issue was noticeable, especially during peak hours when more aircraft routes had to be processed by the CBS Agent. The issue can be explained partly by the development of the simulator in which code is executed sequentially. The computational complexity was also affected by the evaluation of conflicts at every time point.

Furthermore, the repetitive commands sent out by the ATC Agents to correct for the developing dynamics is not in line with how ATC and pilot communicate nowadays. In an operational setting and assuming the current state of technological innovation in an aircraft cockpit, the pilot's multiple commands would increase workload in the cockpit and may lead to safety-related occurrences. Minimum communication between ATC and pilot is preferable for an advanced airport surface movement system. Our last remark is related to the relationship between adaptive behaviour and learning. The use of learning is motivated by the presence of uncertainty in a system. By having an adaptive behaviour, part of this uncertainty is implicitly taken into account, and therefore the evaluation of the contribution of a pure learning method in dealing with uncertainty would not be trivial.

For this research, it was decided to limit the adaptive behaviour of CBS in [5], referred to as Adaptive-CBS from now on, by constraining the frequency by which CBS examines a particular conflict pair. This resembles

the behaviour of the original CBS algorithm, which executes conflict resolution only once, and prior to agent plan execution [6]. To accomplish this, ATC Agents are allocated a memory that enables them to perceive a potential conflict for a more extended period of time and thus preventing the frequent allocation of conflict resolution commands. After the passage of user-defined interval (default set to $20s$), the contents of the memory are erased and the ATC Agent is allowed once again to send commands to the aforementioned conflict pair. The non-adaptive CBS implementation of [5] will be referred to as the Baseline-CBS mechanism for the remainder of the paper.

4 Artificial Neural Networks For Traversal Time Prediction

In our research, we use Artificial Neural Networks (ANNs) to model the traversal time of Aircraft Agents across the links similarly to [12]. ANNs have been widely used in the Machine Learning community and have been shown to approximate any continuous function to a sufficient accuracy [20, 21].

The building block of a neural net is the perceptron. The perceptron has inputs $x_1, x_2, ..., x_i$ weighted by real numbers $w_1, w_2, ..., w_i$ and typically a bias term. Within the perceptron, a summation operation takes place between the weighted inputs and the bias term. If the summation meets a certain condition posed by the activation function ϕ, the output is activated and passed through the network. The use of activation functions makes NNs particularly good at capturing non-linear relationships. ANNs consist of a network of perceptrons stacked on top of each other, and depending on the learning task and available computational resources, one can add multiple layers to a network, resulting in deeper, more complex neural net architectures.

In this study, multiple ANNs are instantiated and allocated to every link eligible to hold a model. The choice to adopt a distributed model architecture comes from a runtime point of view. Mobile Agents are spawned multiple times during the simulation and perform multiple inference calls at a single time point. Using a single unified model, features covering the intrinsic properties of each link would have to be defined, increasing thus the dimensionality of the model.

4.1 Feature Selection

Upon investigating the input data generated after running simulations under a random schedule, it was determined that only past intentions levels did not suffice to explain the target value variance existing in the recorded data. The phenomenon was evident for links of a shorter length where not more than one agent traverses at any given time. Additional features to better capture the dynamics of the Aircraft Agent during its traversal were defined for this purpose. An empirical evaluation of the MAS dynamics determined that the link traversal time is also affected by congestion in neighbouring links, the presence of a turn segment upstream or downstream the agent's path and the allocation of a conflict avoidance command to the Aircraft Agent during its traversla through the link.

Link traversal time is influenced by the inflow of agents at the start of the link and the outflow at a downstream link. To capture this phenomenon in the training data, we use the aircraft density on the neighbouring links as a feature. In the data recording phase, once the agent completes its traversal, the ATC Agents on either edge of the link determine the number of Aircraft Agents taxiing on their respective links and store this information as an attribute to the link in question. In the prediction phase, either during a CBS conflict detection or intention propagation, the instantaneous densities of the neighbouring links are irrelevant since the information required to make the prediction is of a future time-point. Instead, the ATC Agents access the intention timeline of the links and sample the expected intention level at the Aircraft Agent's ETA to that link.

The influence of turns in the traversal time is captured using two categorical features, encoded as binary values: one for entering and another for exiting the link. If the entry or exit occurs via a turn manoeuvre, a value of 1 is set to the respective feature. The ATC Agents acquire this information by accessing the path attribute of the Aircraft Agents. During the prediction phase, the path information is acquired through the Mobile Agent who carries such information.

Conflict resolution commands are generated either by the CBS Agent (re-route and speed) or when the internal logic of the Aircraft Agent determines that a loss of separation is imminent. In order to capture the effect that has in the traversal time, three categorical variables are used to represent each type of executed resolution command. A fourth one is also added to signify the absence of a command. The variables were binarized using one-hot encoding.

Besides the command type, the distance from the link exit at which the Aircraft Agent begins the execution of the command also influences the traversal time. The same goes for the change in speed as a result of a speed command. The latter is captured using two velocity variables, V_1 and V_2. The values assigned to these variables, are based on a set of rules, each representing a particular case. Figure 4 depicts the assignment logic in the data generation phase, in which the Aircraft Agent completes its traversal through the link and its traversal time together with the other relevant feature need to be recorded, for ANN offline training purposes (see Section

6.1. Figure 5, depicts the velocity assignment logic in the case of a CBS conflict detection cycle, in which the Aircraft Agent is completing its taxiing through the link, and the Mobile Agent first has to determine the ETA to the approaching ATC Agent. Lastly, Figure 6 depicts the logic implemented for the prediction of downstream links, where it is not known beforehand whether the Aircraft Agent will execute a conflict avoidance command or not. A downside of the approach is that it works well only for cases where the Aircraft Agent receives a single command during its traversal. If multiple commands are received instead, only the information corresponding to the last executed command will be reflected in the data. Table 1 lists all the features and feature types used as inputs to the ANNs.

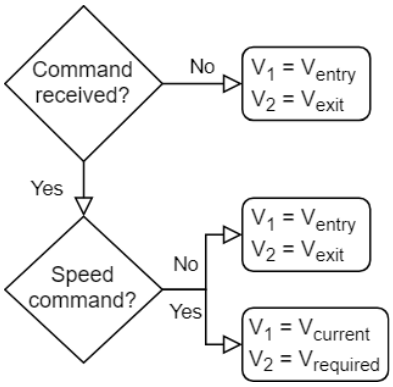

Figure 4: V_1/V_2 assignment logic at the moment the Aircraft Agent completes its traversal and a training sample is recorded.

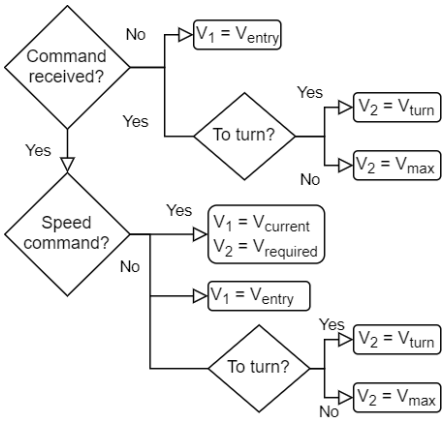

Figure 5: V_1/V_2 assignment logic used in the prediction of the Aircraft Agent's remaining traversal time through a link. The logic is used in the command.

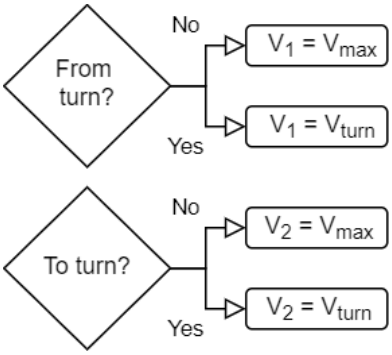

Figure 6: V_1/V_2 assignment logic used in the prediction of the Aircraft Agent's traversal time for the subsequent links of its path.

Table 1: ANN input features and types.

Selected Features	**Type**	**Comments**
Past intention levels (x6)	Continuous	3 minutes, sampled at $30s$ intervals
Neighbouring link densities	Continuous	Amount is link dependent
Aircraft Agent heading to a turn	Categorical	One-hot encoded
Aircraft Agent headed from a turn	Categorical	One-hot encoded
Type of command received	Categorical	One-hot encoded
Distance to next ATC Agent	Continuous	$[m]$
V_1	Continuous	$[m/s]$
V_2	Continuous	$[m/s]$

4.2 Model Properties

Considering the additional layer of complexity the learning mechanism brings to the MAS model, single-layered ANNs have been used. They have previously been argued to be sufficient for approximating any complex nonlinear function [22]. Adopting a single layer architecture also reduces the chances of overfitting, a phenomenon that complex ANN architectures are known to be prone to. An example of a single-layered ANN, in the context of our study, is shown in Figure 7. The models were constructed using the *MLPRegressor* class of the Scikit-Learn Python library [23].

Neural network training is performed via the backpropagation algorithm [24]. At first, the network weights are randomly initialised, and the input values are propagated through the network resulting in a prediction. The prediction is then evaluated against the true value using a cost function. This process is also known as a forward pass. In this study, the square loss function is used as a cost function. Following the forward pass, a backward pass occurs in which the calculated error is fed back to the network, and the weights and biases of the network are updated based on how much they are responsible for the calculated error. Once all training data have been forward and back-propagated, an epoch is completed. The influence of the weights and biases on the calculated error is found using a gradient descent like optimisation algorithm.

We chose to use Adaptive moment estimation (ADAM) [25] as the algorithm to update the weight and biases of the model. ADAM is a computationally efficient optimiser especially suited for non-convex optimisation

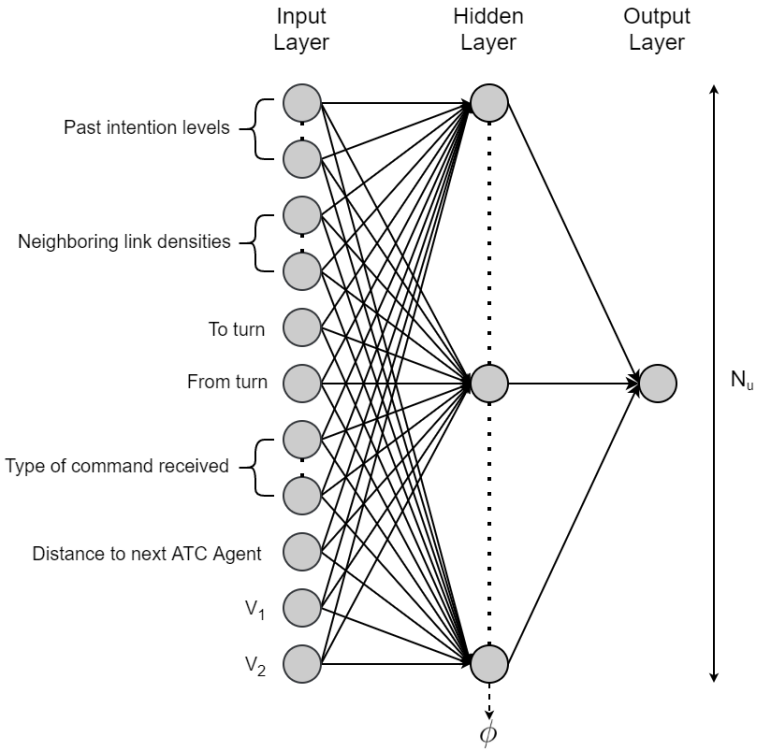

Figure 7: A Single-Layered Artificial Neural Network with N_u hidden layer neurons and activation function ϕ.

problems. The algorithm computes unique adaptive learning rates for different parameters based on estimates of the first and second-order gradients. It combines the notion of momentum optimisation and adaptive learning rate. With momentum optimisation, the past gradients are tracked and used as an accelerating component, meaning that if the gradient is large, then the descent towards the minimum is fastened. Adaptive learning rate adapts the learning rate of the algorithm so that it converges faster to a minimum for dimensions with a larger gradient than for dimensions with smaller. The default value of 0.9 was used for momentum, while the value for the initial learning rate was chosen after performing hyperparameter optimisation. Table 2 lists other training-related parameters.

Table 2: Pre-selected machine learning parameters and their values.

Parameter	**Value**
Number of hidden layers	1
Hidden layer activation function	ReLu
Weight optimiser	ADAM
Number of training epochs	100
Batch size	8

4.3 Hyperparameter Optimisation

With regards to the tuneable parameters, we chose to focus on the number of neurons (N_u), the initial learning rate (η) used by ADAM and the amount of regularisation. Regarding the latter, the *MLPRegressor* class has the option to define levels for the L2 regularisation. Table 3 lists the hyperparameters and their corresponding chosen ranges.

Table 3: Hyperparameters and associated distributions.

Parameter	**Range and Distribution**
N_u	U(10, 60)
Initial η	logU(1^{-4}, 1^{-1})
L2 penalty	logU(1^{-4}, 1^{-1})

In hyperparameter optimisation, the aim is to find parameter values close to optimal by validating their influence on the generalisation performance of the model on a validation set. Common techniques include grid

search and random search [26]. The former is computationally expensive in multidimensional hyperparameter spaces, while random search has shown to outperform grid search in terms of runtime, but at the cost of potentially limited exploration of the hyperparameter space. A technique that overcomes these limitations and has become increasingly popular in ML is Bayesian Optimisation (BO). It allows to configure algorithms without human intervention and has shown to exceed expert-level performance in tuning machine learning models [27]. The objective of BO is to find the minimum of an objective function $f(\boldsymbol{x})$ on some bounded hyperparameter set X.

To do so, it builds a probabilistic model for $f(\boldsymbol{x})$ called a surrogate model and exploits this model to find where in X to next evaluate the function while calibrating both for epistemic and aleatoric uncertainty [27]. BO uses information from previous function evaluations to determine which point of the hyperparameter space to compute next. It can be thought that a learning procedure is running on the background of this process. The more iterations the BO algorithm run, the closer the surrogate model comes to resembling $f(\boldsymbol{x})$. To evaluate which points are to be considered next, an acquisition function a_n is used. The $\boldsymbol{x}_{n+1}$ is chosen is such a way so that a_n is maximised. The pseudo-code for the BO framework [28] is shown in Appendix A of [16]. Table 4 lists the parameters chosen for BO.

Table 4: Parameter selected for BO.

Parameter	**Value**
Surrogate model type	Gradient Boosting Regression Trees (GBRT) [29]
Acquisition call	Expected Improvement (EI) [30]
Number of iterations	400

In order to address potential model performance bias issues caused by splitting the training set between the traditional training and hold-out fashion, a 3-fold cross-validation (CV) strategy was implemented and combined with BO. The Skopt Python library [31] was used for this purpose. The best set of hyperparameters per model was chosen based on the lowest mean CV error. The choice of the performance metric to calculate this is discussed next.

4.4 Regression For Imbalanced Datasets

Uncertainly on the link traversal time arises when bottlenecks occur on taxiways. These bottlenecks make the traversal times longer than average. Such cases are of interest in this research as they can strongly influence CBS's conflict detection and resolution activities. These phenomena, however, do not occur often and are therefore less represented in the data. Consequently, it is expected for the learning algorithm to perform less accurate predictions for these rare cases, as it has been trained to give more importance to the distribution of the nominal and not so important cases. This problem is identified in the literature as the imbalanced dataset problem.

Various modelling strategies for imbalanced domains have been proposed, but most of them relate to classification tasks where the target variable is nominal [32]. For regression tasks where the target variable is continuous, data pre-processing techniques are among the most popular methods to deal with these imbalances. In such methods, the training samples' distribution is altered to steer the learning algorithm towards specific target variable domains. From these techniques, re-sampling strategies offer a friendly way of performing such a distribution change without needing to make changes on the learning algorithm itself. Popular re-sampling strategies are: random under-sampling [33], SMOTER [34] and SMOGN [35]. Random under-sampling is the simplest and less computationally expensive method to use and chosen to use in this research.

Measuring the performance of a learning model on imbalanced datasets is not straightforward. Commonly used performance measures in regression tasks such as mean absolute error and mean squared error are not suitable for problems of this type, as they do not distinguish between nominal and rare cases performance. Instead, they consider the prediction errors equally across the domain of the target variable, using only the magnitude of the error as the decisive factor for the cost of the prediction [34]. To cope with this, we consider the framework of Utility Based Regression (UBR) [36, 37] as a means to differentiate model performance during the phase of hyperparameter tuning.

Utility Based Regression relies on the notions of the relevance function $\phi(y)$ and that of utility. According to [36, 37], $\phi(y)$ is a continuous function that maps the target variable domain into a scale of relevance in which 1 represents highest and 0 lowest relevance. Using a threshold t_R on the relevance values, one can determine the set of the rare and relevant cases D_R and the set of the normal and uninterested cases D_N. In our study, a value of 0.8 was used for t_R.

Two methods have been proposed for creating $\phi(y)$ for a particular target variable distribution. These methods are based on the observation that the notion of relevance is inversely proportional to the target variable probability density function. Information on this density function can be retrieved by box-plot statistics. In

the first method [38], the authors use a sigmoid-based relevance function whose shape is a function of box-plot parameters. The second method [37] uses a piece-wise cubic Hermite interpolation to interpolate a set of relevance function values at user defined control points, $S = \{[y_k, \phi(y_k), \phi^{'(y_k)}]\}_{k=1}^{s}$, where $\phi'(y_k)$ is the first-order relevance derivative of control point k.

The first method was used for this research since it is not dependent on any software package and is faster to implement. The sigmoid-based relevance function is defined as:

$$\phi(\mathbf{Y}) = \frac{1}{1 + e^{-s \cdot (Y - c)}} \tag{3}$$

where c is the center of the sigmoid and s is a parameter which defines its shape.

The second notion of Utility Based Regression, is that of a utility of a prediction. Utility tries to answer whether a prediction led to the correct identification of an extreme type as well as whether that prediction was precise in numeric terms. The author at [37] proposed the following definition for the utility of a particular prediction:

$$\begin{aligned} U_{\phi}^{p}(\hat{y}, y) &= B_{\phi}(\hat{y}, y) - C_{\phi}^{p}(\hat{y}, y) \\ &= \phi(y) \cdot (1 - \Gamma_B(\hat{y}, y)) - \phi^{p}(\hat{y}, y) \cdot \Gamma_C(\hat{y}, y) \end{aligned} \tag{4}$$

where $B_{\phi}(\hat{y}, y), \Gamma_B(\hat{y}, y)), \Gamma_C(\hat{y}, y), C_{\phi}^{p}(\hat{y}, y)$ are functions related to the costs and benefits of predictions.

The above definition is used to define recall and precision functions for regression problems. In this research we use the definitions provided in [39], shown in Equations 5 and 6, which can be combined into the F_1-measure [40].

$$recall = \frac{\sum\limits_{\phi(y_i) > t_R} (1 + u_i)}{\sum\limits_{\phi(y_i) > t_R} (1 + \phi(y_i)} \tag{5}$$

$$precision = \frac{\sum\limits_{\phi(\hat{y}_i) > t_R} (1 + u_i)}{\sum\limits_{\phi(\hat{y}_i) > t_R} (1 + \phi(\hat{y}_i)} \tag{6}$$

5 Verification and Validation

Verification and validation were performed according to the recommended practices in [41, 42]. The face validity of the MAS model was assessed iteratively using animation, and immersive assessment techniques [41]. A graphical user interface was used during animation assessment to display the aircraft motions within the taxiway system and how the controller agents' actions influenced that. The observed animated traffic patterns were compared to the animated output of the original MAS model and were found to have similar behaviour. Immersive assessment techniques were utilised to check the behaviour of particular agents and their interactions with other agents and the environment. Examples of such techniques include displaying the intention level database of ATC Agents, the filtered intention timeline of the taxiway links and information regarding the status of the CBS Agent, such as the dMAS counter or details regarding the predicted conflict. In addition, historical data validation in the form of k-fold cross-validation was performed during the hyperparameter tuning process of the neural networks. Computerised model verification was carried out throughout the validation procedure of the MAS model by performing frequent unit testing and code debugging. This ensured the sound implementation of the conceptual model into code.

6 Experimental Analysis

This section presents the experimental analysis of the dMAS-based CBS mechanism. Section 6.1 describes the offline data generation process for the purposes of offline learning. The outcome of the hyperparameter optimisation is discussed in Section 6.2. While in Section 6.3, the results of our experiments are presented and discussed. All code during model development and experimental analysis was developed in Python 3.7.

6.1 Data generation

Semi-random flight schedules were created in order to facilitate the training of the machine learning models. The schedules are based on the average hourly traffic counts of an 8-day real-world dataset and consist of departure/arrival times and origin/destination pairs. The real-world dataset was used for performance analysis later on. In order to generate the schedules, several assumptions were made. Firstly, it is assumed that the arrival rate of the aircraft to the airport system (arriving and departing separated) follows a Poisson distribution,

and consequently, the inter-arrival times are computed through the inverse CDF technique, given in Equation 7.

$$F_X^{-1}(t) = -\frac{ln(1-t)}{\lambda} \tag{7}$$

The mean arrival rate per unit hour, λ, is calculated by dividing each day into 24 intervals and averaging the traffic counts per mode of traffic across all days. The number of runway reconfigurations is assumed to be constant and equal to 18 for each day, which is extracted from historical data. The time interval between the configuration is also assumed to be constant. Furthermore, we distinguish between two modes of operations: a departure peak and an arrival peak. For each peak, a 2+1 runway configuration is assumed. The choice of the runway configuration is based on statistics of the most frequent use of combinations at AAS. Tables 5, 6 list the runway mode of operations considered. Lastly, gates are randomly assigned, and in case of arriving traffic, the entry to the taxiway system is biased to runway exits closer to the runway end.

Table 5: Runway configurations assumed for arrival peaks.

Inbound	Outbound
27, 18R	24
18R, 18C	24
06, 36R	36C

Table 6: Runway configurations assumed for departure peaks.

Inbound	Outbound
36C	36L, 09
06	36L, 09
18R	18C, 18L

Using the semi-random schedules, a sequence of simulations was performed to generate training data for the links chosen to participate in the learning task. During the simulations, the dMAS mechanism was switched on, allowing intentions to be propagated in the environment. Conflict prediction and resolution activities were performed using the baseline CBS algorithm. At the end of each link traversal, the relevant ATC Agent recorded the $(\mathbf{X}, y)$ sample and stored it in a link specific database. In total, 700 simulations were performed, each under a different seed, allowing thus for multiple types of traffic patterns to occur. Night traffic, between 23:00 and 05:00, was excluded from the data generation process as only a few movements take place during this time. After the simulations, the recorded data were aggregated across all days and training data sets for each link was formed. In total, data for 176 out of 265 links ($\approx 66\%$) was recorded. Subsequently, links with a low number of samples were discarded as they were not used often by the aircraft, and their contribution to the overall performance would be minimal. A prediction query was always set to return the average traversal time for these links, which is calculated based on the recorded data and updated online at the end of every traversal. After the datasets were formed, an ANN model was allocated to each of the remaining 158 links. The outcome of the hyperparameter tuning process is discussed next.

6.2 Hyperparameter Optimisation

Four experiments were performed during the hyperparameter optimisation stage. In the first experiment, all the training data were used, while in the remaining experiments, undersampling of ratios 200, 300 and 400 were applied respectively and according to the principles of the UBR framework. Before the optimisation, each dataset was split in an 80%-20% fashion into training and test sets. The training set was used to determine the best set of hyperparameters by performing a Bayesian Optimisation coupled with a 3-fold CV. The optimisation was initialised by randomly probing 10 points from the hyperparameter space and was limited to 400 iterations. The hyperparameter combination that returned the lowest F_1 score was then selected. For the computations, an Intel Xeon CPU was used, consisting of 128 2.4GHz cores. Upon completion of the optimisation, the generalisation performance of the chosen model was tested against a baseline prediction on the test set. The baseline is simply a naive prediction that returns the average target value of the training data. If a model fails to generalise better than its naive counterpart, it is discarded and replaced by naive logic. In this way, we ensure that the models chosen are guaranteed to match or surpass the generalisation performance of the average prediction, and if selected, they at least have demonstrated the ability to learn.

Table 7 presents the Mean Absolute Error (MAE) and F_1 values of each prediction model when tested on the test set, under different sampling strategies. In effect, we observe that ANNs predict better than the average for all sampling strategies, which is confirmed by the lower MAE and higher F_1 scores. The associated error distributions are shown in Figure 8. The results indicate that when applying undersampling, the generalisation performance of the ANNs deteriorates as opposed to the performance of the naive prediction, which remains more or less constant. This outcome is not in line with the claims made in Section 4.4. In our context, reducing the normal cases in the datasets by applying undersampling seems to have a negative impact on the predictions. Specifically, without applying any sampling, at least 10% more ANNs had a better generalisation performance than the naive prediction resulting thus in a higher participation of learning models in the planning mechanism. In Appendix B of [16] the links which were allocated an ANN under each sampling strategy are shown. In

total, 134, 135 and 139 ANNs were allocated for the experiments with undersampling ratios of 200, 300 and 400 respectively.

Table 7: Model test set evaluation results.

Sampling strategy	**Mean** MAE_{ANN} (s)	**Mean** MAE_{Naive} (s)	**Mean** $F_{1_{ANN}}$ **score**	**Mean** $F_{1_{Naive}}$ **score**	**% of models with** $F_{1_{ANN}} > F_{1_{Naive}}$
No sampling	**0.79**	3.50	**0.71**	0.11	**97**
U(200)	1.63	3.33	0.69	0.11	85
U(300)	1.37	3.51	0.69	0.11	85
U(400)	1.17	3.51	0.70	0.11	87

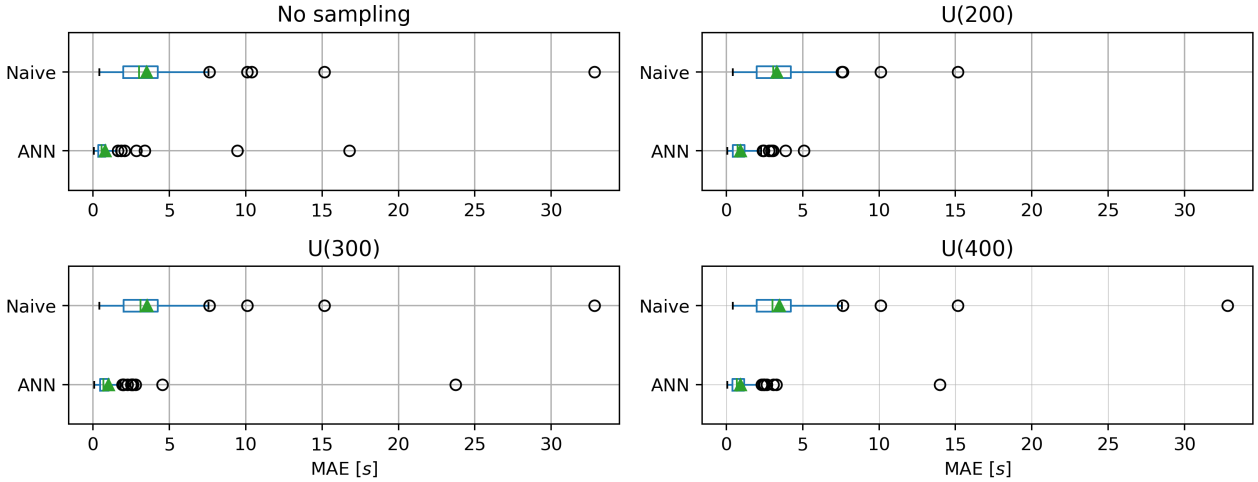

Figure 8: Distribution of the MAE across all models per experiment type.

Regarding the choice of hyperparameters, Figure 9 depicts the respective probability density function estimates for the models developed under each sampling strategy. The distributions chosen when defining the hyperparameter spaces are also shown for reference. For the case of the number of neurons, the majority of the models resulted in a high number of neurons, between 50 to 60. The distributions of values for the L2 regularisation are shown to follow the log-normal nature of the pre-defined distribution, with the majority of values belonging to the 0.001-0.01 interval. The same goes for the learning rate values for which the highest percentage falls in the 0.001-0.02 interval.

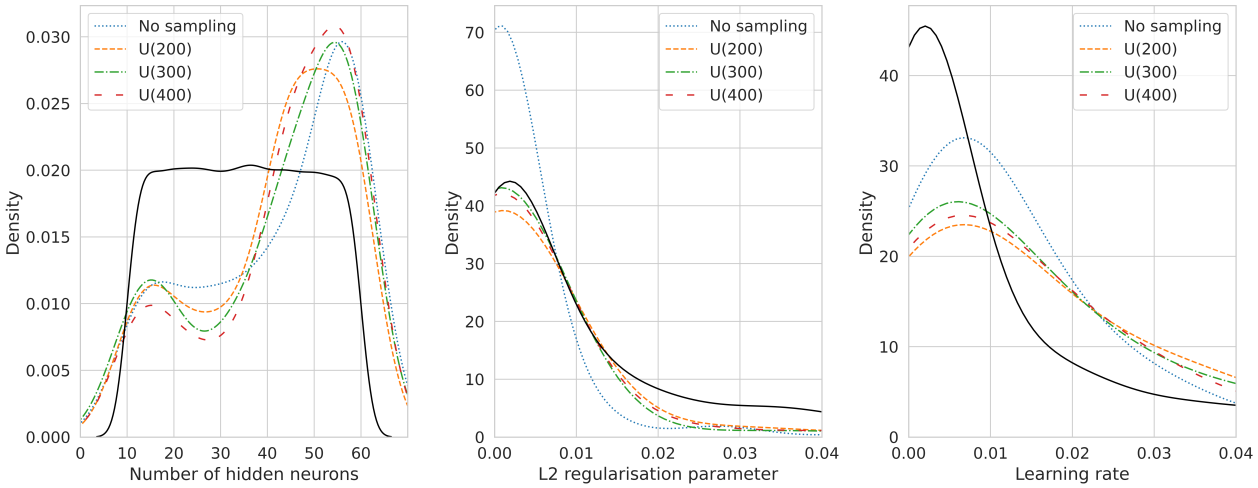

Figure 9: Kernel density plots of the best set of hyperparameters for models trained on not sampled datasets. The black lines correspond to the shapes of the pre-defined hyperparameter distributions.

6.3 Results

The influence of the learning mechanisms in the airport surface movement operations is assessed from two viewpoints. The first viewpoint focuses on the overall performance of the agent-based model and aims at determining whether the addition of a learning-based mechanism improves commonly used Key Performance Indicators (KPIs). In the second viewpoint, the focus is shifted to comparing the conflict detection abilities of the considered mechanisms, answering thus the question of whether the ML predictions result in a better conflict identification than the analytical method.

6.3.1 Aggregated Simulation Results

At first, the MAS model was tested under an eight-day real-world flight schedule of AAS. The schedule was extracted from an ADS-B data analysis performed in [4], consists of origin and destination (O/D) pairs, aircraft spawn times and distinguishes the flights between inbound and outbound. The various O/D pairs present in the schedule allow for a wide range of traffic patterns and capture operational scenarios such as runway reconfigurations and inbound/outbound peaks. Performance analysis was carried out at an aircraft level using a set of KPIs. These are the aircraft taxi time duration (AC_{tt}), the distance covered during taxiing (AC_{td}) and the average taxi speed (AC_{ts}).

To determine the appropriate statistical test to use, the Shapiro-Wilk and the D'Agostino's K2 tests were first used to conclude on the normality of the obtained samples. The tests showed that the samples were not normally distributed, and therefore, the non-parametric Vargha-Delaney A-test [43] was chosen to compare the differences between the distributions. It is an effect size test commonly used to compare the results of MAS models. In contrast to other tests that measure statistical significance, the A-test provides a measure for the scientific significance of the differences in the distributions of two samples. It determines a value between 0 and 1.0 with 0.6, for example, denoting the probability that a randomly selected sample from one distribution has a higher value from the respective sample of its paired distribution. The A-test values are typically classified as showing a small, medium or large effect due to the differences in sample values, depending on whether their value is over 0.56, 0.64 and 0.71, respectively. For values below 0.5, the same intervals are applied.

Table 8 lists the average simulation results of all days in the real-world flight schedule for the Baseline, dMAS and dMAS-U(300) CBS variants. The remaining dMAS variants failed to successfully resolve all conflicts in the flight schedule and were discarded from this analysis. The distributions of the KPIs for each flight day and mechanism type can be found in Appendix B of [16]. Regarding the two dMAS based mechanisms that were able to successfully resolve all conflicts, we observe that no differences exist between the values of their KPIs and those of the Baseline mechanism. This is also confirmed by the A-test values being equal or close to 0.50. These results suggest that the application of dMAS or dMAS-U(300) do not seem to improve nor to deteriorate the performance of the airport surface movement operations. The invariance in the results can be attributed to the limited time during which dMAS was active. This was driven by the number of conflicts that took place during these intervals. As mentioned before, dMAS CBS becomes active for 20 seconds after the Baseline-CBS has detected a conflict. The dMAS time counter is then updated for every conflict detected by dMAS-CBS within that time period. The overall traffic handled by dMAS-CBS during these intervals may not have been sufficient to produce significant changes in the dynamics of the surface movements, and hence its effect is not evident at a global level. Furthermore, this outcome provides no conclusion regarding the effectiveness of the learning mechanism at detecting conflicts. A closer analysis, specifically at the conflict detection level, was conducted for this purpose and is described next.

Table 8: MAS average taxi time (AC_{tt}), taxi distance (AC_{td}) and taxi speed (AC_{ts}) of flights on all days.

CBS Mechanism	AC_{tt} (min/**flight**)	AC_{tt} **A-test Value**	AC_{td} (km/**flight**)	AC_{td} **A-test Value**	AC_{ts} ($\frac{m}{s}$/**flight**)	AC_{ts} **A-test Value**
Baseline-CBS	$\mu = 5.62$ $\sigma = 3.24$	-	$\mu = 3.83$ $\sigma = 2.55$	-	$\mu = 10.86$ $\sigma = 2.11$	-
dMAS-CBS	$\mu = 5.63$ $\sigma = 3.25$	0.50	$\mu = 3.83$ $\sigma = 2.55$	0.50	$\mu = 10.85$ $\sigma = 2.10$	0.49
dMAS-U(300)-CBS	$\mu = 5.64$ $\sigma = 3.25$	0.50	$\mu = 3.83$ $\sigma = 2.55$	0.50	$\mu = 10.85$ $\sigma = 2.10$	0.49

6.3.2 Conflict Prediction Analysis

In the second type of analysis, the conflict detection abilities of the (dMAS)-CBS variants were evaluated under three operational scenarios: departure peaks, arrival peaks and periods with frequent runway reconfigurations. The comparison is conducted by measuring the precision of the conflict detection capabilities of each method in the absence of the resolution component of CBS. To accomplish this, the predicted conflict timepoints are recorded and compared with the actual times the aircraft agents reach the predicted conflict location. For such measurements, the CBS conflict resolution mechanism is deactivated, forcing the Aircraft Agents to continue their taxi even if a conflict is imminent. This alteration is applied only for conflicts of crossing nature. The resolution of head-on conflicts is not excluded from our analysis because these conflicts usually cause severe bottlenecks, preventing thus the Aircraft Agents from reaching the predicted conflict point. This would hinder any effort towards measuring the actual time of arrival for these agents. In addition, head-on conflicts account, on average, for less than 10% of the total amount of conflicts detected.

A downside of such a measurement procedure is the introduction of a measurement bias caused by the self-separation logic of the Aircraft Agents. To illustrate this, consider the example of having a conflict predicted correctly. This would mean that as the two Aircraft Agents approach the conflict node, and in the absence of a CBS resolution component, their separation would be lost, forcing either one to slow down, shifting thus the actual time of arrival forward in time. A solution to this is to deactivate the self-separation logic of the aircraft. However, this is less realistic and would imply that conflicts that were not detected by CBS would not be resolved locally. Therefore, it was decided to stay with the self-separation activated as it is a principle that applies for all experiments regardless of the type of mechanism used. Once the Aircraft Agents reach the predicted conflict point, their actual time of arrival (ATA) is recorded and compared to the ETA of the prediction mechanism.

The operational scenarios were extracted by analysing the properties of each flight day in the real-world flight schedule. The analysis showed that flight days 04/05 and 13/05 had a high number of movements during an arrival or departure peak period and also had periods with a high number of runway reconfigurations. Tables 9 and 10 list the flight schedule details concerning the four operational scenarios.

Table 9: Flight schedule details for Scenario A (04/05) and Scenario B (13/05).

Scenario	**Time interval**	**Inbound traffic count**	**Outbound traffic count**	**Inbound RWYs**	**Outbound RWYs**
A (Inbound Peak)	07:00 - 09:00	80	44	R18C, R18R	R18L, R24
	12:00 - 14:00	65	53	R18C, R18R	R18L, R24
B (Outbound Peak)	09:00 - 11:00	35	78	R06, R36R	R36C, R36L
	20:00 - 22:00	19	75	R06, R36R	R36C, R36L

Table 10: Flight schedule details for Scenario C (04/05) and Scenario D (13/05).

Scenario	**Time interval**	**Inbound traffic count**	**Outbound traffic count**	**RMOs (Inbound, Outbound)**
C	16:00 - 18:00	38	41	1. R18C + R18R, R36L 2. R18R, R18L + R24 3. R18R, R24 4. R18R + R27, R36L 5. R06 + R27, R36L
D	11:00 - 13:00	46	60	1. R36R, R36L 2. R36R + R06R, R36L 3. R36R + R06, R36L + R36C 4. R36R, R36L + R36C 5. R36R + R06, R36L + R36C

The measurements conducted focus on the following: the number of conflict avoidance commands the Aircraft Agents had to adhere to, the conflict prediction error for conflicts that occurred during the intervals in which dMAS-reinforced CBS was active and lastly, the precision of the predictions and their relation with the previously mentioned KPIs.

Number of conflict avoidance commands (CBS vs Visual Separation)

Table 11 lists the average amount of conflict avoidance commands the Aircraft Agents had to adhere to during each type of simulation. The commands are distinguished between CBS and Visual Separation (VS). The latter are executed by the Aircraft Agents when their internal logic determines that their separation with another aircraft is lost. This measurement, and specifically the amount of VS commands executed, provides insight regarding the effectiveness of CBS at providing separation to the aircraft. The higher the VS commands executed, the more future conflicts CBS failed to detect. Table 12 lists the A-test values of all mechanisms when compared to the Adaptive CBS and the values of the dMAS variants when compared to the Baseline-CBS. Note that the dMAS variants become active only during an uncertainty window. For timepoints outside this window, the Baseline CBS becomes the active mechanism.

Table 11: Average number of conflict avoidance commands allocated per CBS mechanism in each operational scenario.

CBS Mechanism	**Scenario A**		**Scenario B**		**Scenario C**		**Scenario D**	
	CBS	*VS*	*CBS*	*VS*	*CBS*	*VS*	*CBS*	*VS*
Adaptive-CBS	13.92	6.02	12.58	**9.53**	11.0	4.6	7.88	6.79
Baseline-CBS	2.10	6.33	1.03	9.64	1.53	3.6	0.65	6.79
dMAS-CBS	**1.22**	**5.81**	0.95	10.32	**1.00**	4.27	0.7	6.79
dMAS-U(200)-CBS	**-**	**-**	0.90	9.75	1.21	3.44	**0.56**	6.79
dMAS-U(300)-CBS	1.47	6.51	**0.85**	10.32	**1.00**	**2.28**	0.64	6.79
dMAS-U(400)-CBS	1.45	6.11	0.93	10.39	1.10	3.71	0.58	6.79

Table 12: A-test values for Scenarios A and B, between Adaptive-CBS and non-adaptive variants (left-inner column) and between the baseline CBS and the dMAS variants (right-inner column).

CBS Mechanism	**Scenario A**				**Scenario B**				**Scenario C**				**Scenario D**			
	CBS		*VS*		*CBS*		*VS*		*CBS*		*VS*		*CBS*		*VS*	
Baseline-CBS	0.38	-	0.51	-	0.38	-	0.50	-	0.42	-	0.49	-	0.40	-	0.50	-
dMAS-CBS	0.35	0.45	0.50	0.49	0.38	0.49	0.51	0.51	0.41	0.47	0.51	0.51	0.41	0.50	0.50	0.50
dMAS-U(200)-CBS	-	-	-	-	0.38	0.49	0.50	0.50	0.41	0.48	0.49	0.49	0.40	0.49	0.50	0.50
dMAS-U(300)-CBS	0.35	0.46	0.51	0.50	0.37	0.48	0.51	0.51	0.41	0.48	0.48	0.48	0.40	0.50	0.50	0.50
dMAS-U(400)-CBS	0.36	0.46	0.51	0.50	0.37	0.49	0.51	0.50	0.41	0.49	0.49	0.50	0.40	0.50	0.50	0.50

It can be seen that the number of CBS commands has decreased substantially when using the non-adaptive variants. This result is expected since the non-adaptive variants execute the conflict detection and resolution cycle less frequently than the adaptive case. The effect of this change is small to medium, according to the A-test values. In addition, it is seen that the decrease in the CBS commands does not negatively impact the amount of the VS commands being executed. All changes observed between the adaptive and the non-adaptive variants and between the dMAS and its variants were found to have a negligible effect. For Scenario A, a bottleneck that occurred when using dMAS-U(200) prevented the simulation from completing, and therefore data was not recorded. With respect to the allocated amount of CBS commands, the lowest amount was found for the dMAS variant with a reduction of 42% compared to the Baseline. In Scenario B, the dMAS-U(300) variant allocated the least CBS commands on average, while the slight increase on the VS commands allocated by the non-adaptive variants compared to the Adaptive variant was found to have a negligible effect. In Scenario C, the dMAS and dMAS-U(300) variants had a similar performance regarding the allocated CBS commands, but the latter was found to contribute to fewer aircraft VS commands. In Scenario D, no significant differences were found between the non-adaptive variants. The VS commands remained constant across all experiments.

Conflict prediction error

The conflict prediction error is calculated from the absolute differences between the estimated times and actual time of arrivals. We distinguish between the error present in the prediction of the commanded, AC_c, or the other aircraft in a conflict pair, AC_o (Equations 8 and 9). In addition, we define a measure for the CBS error, which is given in Equation 10. The latter is used to determine whether CBS lead to the correct identification of a conflict. Error values less or equal to T_{window} correspond to conflicts that actually took place. Tables 13, 14 list the mechanisms' absolute prediction error retrieved under Scenarios A and C respectively. The tables corresponding to Scenarios B and D can be found in Appendix B of [16]. Furthermore, in Figure 10 the error distributions for the Adaptive, Non-adaptive and dMAS CBS variants under Scenario A are shown. The remaining mechanisms and scenarios can as well be found in [16].

$$AC_{c_{AE}} = |ETA_{cmd} - ATA_{cmd}| \tag{8}$$

$$AC_{o_{AE}} = |ETA_{other} - ATA_{other}| \tag{9}$$

$$CBS_{AE} = ||ETA_{cmd} - ETA_{other}| - |ATA_{cmd} - ATA_{other}|| \tag{10}$$

As mentioned above, these results concern the conflicts detected during the timepoints in which dMAS-CBS was active. Regarding the non-adaptive variants, it is important to note that because their underlying detection mechanism is different, the number of conflicts detected by either one will vary. A conflict predicted by one mechanism does not necessarily mean that it will be predicted by the other. More precisely, between the Baseline and the dMAS variants, the largest differences were found for dMAS-CBS in Scenarios A and C where 69% and 64% fewer conflicts were detected respectively, dMAS-U(300) in Scenario B corresponding to a reduction of 33% and dMAS-U(200) in Scenario D for a reduction of 41%. We see that in all scenarios, the dMAS variants were more conservative than the Baseline when making conflict predictions. Regarding the mean and standard deviation of the prediction error, we cannot make an immediate conclusion, as the sample size of each mechanism is different. Looking at the individual distributions, however, we notice a higher uncertainty in the predictions of the Baseline mechanism, especially during periods with an increased number of runway reconfigurations. Many of the predictions of the Adaptive-CBS, on the other hand, are outliers. Despite that, its average error in all scenarios is maintained at low levels. The latter can be explained by the fact that conflict prediction for a particular conflict pair in Adaptive-CBS is performed throughout the travel of that pair to their predicted conflict location. The further away they both are, the more uncertainty is present, which is seen as an outlier in the data. As the two aircraft approach each other, the underlying uncertainty in their paths decreases, making the Adaptive-CBS predictions better.

Table 13: Statistics of the absolute prediction error for predictions made in Scenario A.

CBS Mechanism	$N_{conflicts}$	$AC_{c_{AE}}$ $[s]$	$AC_{o_{AE}}$ $[s]$	CBS_{AE} $[s]$
Adaptive-CBS	2692	$\mu = 30.20$ $\sigma = 40.44$	$\mu = 47.20$ $\sigma = 59.34$	$\mu = 52.99$ $\sigma = 63.43$
Baseline-CBS	267	$\mu = 61.43$ $\sigma = 70.90$	$\mu = 86.58$ $\sigma = 98.84$	$\mu = 102.09$ $\sigma = 107.27$
dMAS-CBS	82	$\mu = 32.56$ $\sigma = 27.52$	$\mu = 49.44$ $\sigma = 39.26$	$\mu = 24.66$ $\sigma = 29.19$
dMAS-U(200)-CBS	-	-	-	-
dMAS-U(300)-CBS	143	$\mu = 39.03$ $\sigma = 27.08$	$\mu = 51.45$ $\sigma = 34.65$	$\mu = 32.92$ $\sigma = 31.16$
dMAS-U(400)-CBS	110	$\mu = 58.36$ $\sigma = 85.53$	$\mu = 62.21$ $\sigma = 72.08$	$\mu = 66.01$ $\sigma = 116.16$

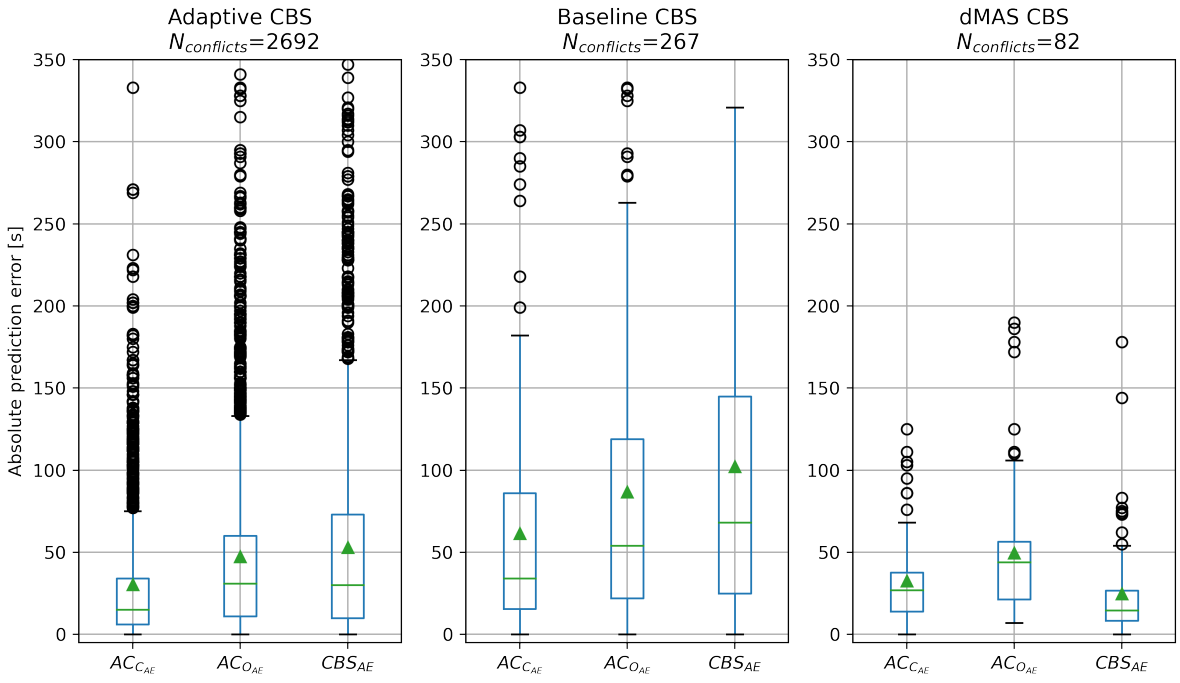

Figure 10: Conflict prediction error distributions of Adaptive, Baseline and dMAS CBS in Scenario A.

Moreover, we see that in Scenarios C (Table 14) and D (Appendix B [16]), all mechanisms detect fewer conflicts compared to Scenarios A and B. This is due to the lower traffic levels existing during runway reconfigurations as the airport operates in a reduced capacity, having thus less coupled traffic. The results also suggest that the prediction errors for the AC_o are higher than for the AC_c. This is observed across all mechanisms and scenarios. The result is counterintuitive since AC_c is the aircraft that is the furthest away from the conflict location, and we would expect that to be the one that most deviates from the predicted conflict time. A possible explanation could be that the conflict prediction module of CBS only accounts only for conflict pairs and not for higher-order dependencies between aircraft. This implies that after a command has been generated, the other aircraft may not be entirely decoupled from existing traffic and is subject to CBS's handling at a later timepoint. Further analysis, however, needs to be performed to support this claim.

Table 14: The mean and standard deviation of the absolute prediction error for predictions made in Scenario C.

CBS Variant	$N_{conflicts}$	$AC_{c_{AE}}$ $[s]$	$AC_{o_{AE}}$ $[s]$	CBS_{AE} $[s]$
Adaptive CBS	754	$\mu = 33.74$ $\sigma = 40.61$	$\mu = 42.96$ $\sigma = 54.54$	$\mu = 52.17$ $\sigma = 66.12$
Baseline CBS	58	$\mu = 61.34$ $\sigma = 66.87$	$\mu = 100.19$ $\sigma = 99.28$	$\mu = 103.78$ $\sigma = 94.68$
dMAS-CBS	21	$\mu = 55.38$ $\sigma = 35.46$	$\mu = 68.81$ $\sigma = 50.53$	$\mu = 29.33$ $\sigma = 32.16$
dMAS-U(200)-CBS	35	$\mu = 61.80$ $\sigma = 40.68$	$\mu = 83.00$ $\sigma = 64.03$	$\mu = 35.74$ $\sigma = 45.13$
dMAS-U(300)-CBS	21	$\mu = 47.05$ $\sigma = 45.54$	$\mu = 58.43$ $\sigma = 47.63$	$\mu = 56.24$ $\sigma = 62.36$
dMAS-U(400)-CBS	24	$\mu = 50.21$ $\sigma = 35.39$	$\mu = 61.58$ $\sigma = 49.65$	$\mu = 40.00$ $\sigma = 34.39$

In Table 15, the mean and standard deviation of CBS_{AE} for the Baseline and dMAS-CBS on conflict predictions that are common to both mechanisms are presented. These concern predictions that took place at the same timepoint and cover the same conflict pair and location. For Scenario A, the $1s$ increase in the mean error was found to have no effect or statistical significance. This also applies to Scenarios B and D, where a decrease of approximately $2s$ and an increase of $1s$ were found, respectively. The only significant result was found for Scenario C, where the average error of dMAS-CBS was $13.5s$ lower for a sample of 9 conflicts. The A-test value of 0.43 signifies that the effect due to dMAS-CBS is small. In Table 16 the prediction statistics for the commonly identified conflicts between the dMAS variants are shown. For Scenarios A and B, any variation in the CBS_{AE} had a negligible effect according to the A-test values, whereas in Scenario C, the dMAS and dMAS-U(200) mechanisms had a better prediction performance than the remaining mechanisms. Lastly, in Scenario D, all dMAS based variants showed better predicting abilities than dMAS. The greatest improvement was observed for dMAS-U(200), equal to $12s$.

Table 15: CBS_{AE} mean and standard deviation for common conflict pairs between the Baseline and dMAS CBS.

	Scenario A $N_{conflicts} = 34$		**Scenario B** $N_{conflicts} = 28$		**Scenario C** $N_{conflicts} = 9$		**Scenario D** $N_{conflicts} = 10$	
Baseline CBS	$\mu = 24.44$ $\sigma = 32.65$	-	$\mu = 44.68$ $\sigma = 52.87$	-	$\mu = 51.56$ $\sigma = 53.14$	-	$\mu = 30.00$ $\sigma = 39.19$	-
dMAS-CBS	$\mu = 25.41$ $\sigma = 30.74$	A-test = 0.55	$\mu = 42.39$ $\sigma = 49.81$	A-test = 0.48	$\mu =$ **38.00** $\sigma = 44.77$	A-test = **0.43**	$\mu = 31.00$ $\sigma = 37.44$	A-test = 0.52

Table 16: CBS_{AE} mean and standard deviation for common conflict pairs between the dMAS variants.

CBS Mechanism	**Scenario A** $N_{conflicts} = 27$		**Scenario B** $N_{conflicts} = 25$		**Scenario C** $N_{conflicts} = 7$		**Scenario D** $N_{conflicts} = 7$	
dMAS-CBS	$\mu = 17.93$ $\sigma = 12.75$	-	$\mu = 31.28$ $\sigma = 33.28$	-	$\mu = 34.71$ $\sigma = 49.14$	-	$\mu = 28.14$ $\sigma = 24.10$	-
dMAS-U(200)-CBS	$\mu = 17.22$ $\sigma = 13.66$	A-test = 0.47	$\mu = 32.60$ $\sigma = 34.60$	A-test = 0.49	$\mu = 34.86$ $\sigma = 49.27$	A-test = 0.51	$\mu =$ **16.43** $\sigma = 26.70$	A-test = 0.19
dMAS-U(300)-CBS	$\mu = 20.22$ $\sigma = 15.90$	A-test = 0.54	$\mu = 32.36$ $\sigma = 34.70$	A-test = 0.49	$\mu = 57.14$ $\sigma = 64.03$	A-test = 0.66	$\mu =$ **20.14** $\sigma = 27.54$	A-test = 0.29
dMAS-U(400)-CBS	$\mu = 20.04$ $\sigma = 14.32$	A-test = 0.54	$\mu = 31.56$ $\sigma = 33.01$	A-test = 0.50	$\mu = 46.86$ $\sigma = 50.89$	A-test = 0.60	$\mu =$ **18.71** $\sigma = 24.56$	A-test = 0.19

Conflict prediction precision

The suitability of the proposed mechanisms is further assessed by looking at the amount of true positive (TP) and false-positive predictions (FP) they generate. A true positive prediction is defined as one in which the absolute CBS error (Equation 10) is less than or equal to the conflict time window, T_{window}. We then use the measure of *Precision* to assess the mechanisms. *Precision* is defined as the ratio of the true positives over all the positively identified instances (Equation 11). The more precise a mechanism is, the less the probability of predicting false positives, also known as Type I error. In our operational setting, this would translate into having the Aircraft Agents receiving fewer erroneous commands by the ATC Agents. As we saw earlier in Section 4.4, it is common practice to combine *Precision* with the measure of *Recall*, which takes into account the false negatives predictions (Type II error). However, in our simulations, we do not identify false negative predictions as we lack a reference that declares a conflict as true. The only conflict information we capture is based on the mechanisms' conflict detection principles, meaning that the only validation that can be performed is only that of an identified conflict. However, we know that a falsely unidentified conflict will trigger the visual separation logic of the Aircraft Agents at a later timepoint, causing them to slow down or change paths. Tables 18 to 21 present the precision of each mechanism and scenario alongside the previously mentioned KPIs.

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

Table 17: CBS_{AE} statistics for common conflict pairs between the dMAS variants.

	Adaptive	**Baseline**	**dMAS**	**dMAS-U(200)**	**dMAS-U(300)**	**dMAS-U(400)**
TP / FP	940 / 1752	58 / 210	42 / 40	41 / 60	46 / 97	45 / 65
Precision	0.35	0.22	0.51	0.41	0.32	0.41
$\overline{AC_{tt}}$ $[min]$	6.71	7.14	6.35	6.99	6.92	6.44
$\overline{AC_{td}}$ $[km]$	4.77	5.12	4.75	4.95	4.75	4.73
$\overline{AC_{ts}}$ $[m/s]$	11.31	11.58	11.09	11.20	10.97	11.00

Table 18: Precision and KPI values of all mechanisms in Scenario A.

Table 19: Precision and KPI values of all mechanisms in Scenario B.

	Adaptive	**Baseline**	**dMAS**	**dMAS-U(200)**	**dMAS-U(300)**	**dMAS-U(400)**
TP / FP	975 / 1833	27 / 90	43 / 65	26/71	13 / 66	26 / 79
Precision	0.35	0.23	0.40	0.27	0.16	0.25
$\overline{AC_{tt}}$ $[min]$	6.39	7.07	6.91	6.66	6.98	6.67
$\overline{AC_{td}}$ $[km]$	4.07	4.59	4.42	4.27	4.59	4.32
$\overline{AC_{ts}}$ $[m/s]$	10.46	10.71	10.69	10.58	10.84	10.66

Table 20: Precision and KPI values of all mechanisms in Scenario C.

	Adaptive	**Baseline**	**dMAS**	**dMAS-U(200)**	**dMAS-U(300)**	**dMAS-U(400)**
TP / FP	172 / 582	11 / 47	7 /14	16 / 19	0 / 21	6 / 18
Precision	0.23	0.19	0.33	0.46	0.00	0.25
$\overline{AC_{tt}}$ $[min]$	8.44	8.85	8.38	8.34	9.86	9.53
$\overline{AC_{td}}$ $[km]$	6.04	6.48	5.94	5.82	7.24	6.07
$\overline{AC_{ts}}$ $[m/s]$	11.49	11.96	11.09	11.12	11.65	11.19

Table 21: Precision and KPI values of all mechanisms in Scenario D.

	Adaptive	**Baseline**	**dMAS**	**dMAS-U(200)**	**dMAS-U(300)**	**dMAS-U(400)**
TP / FP	306 / 475	5 /24	6 / 30	8 / 9	8 / 18	9 / 12
Precision	0.39	0.17	0.17	0.40	0.31	0.43
$\overline{AC_{tt}}$ $[min]$	5.62	6.31	6.41	6.53	6.66	6.29
$\overline{AC_{td}}$ $[km]$	3.64	4.16	4.15	4.23	4.39	4.18
$\overline{AC_{ts}}$ $[m/s]$	10.50	10.83	10.47	10.35	10.83	10.63

Tables 18, 19, 20 and 21 list the Precision and the associated KPIs of the mechanisms, for Scenarios A, B, C and D respectively. Overall, the dMAS variants are shown to be more precise when characterising conflicts. In scenarios with peak traffic, pure dMAS is the most precise mechanism of all. In Scenario A, dMAS scored a precision of 51% compared to 35% and 22% that the Adaptive and Baseline CBS scored. The result is accompanied by a reduction in the mean values of all three KPIs. The decrease, however, has a negligible effect according to the associated A-test values. Furthermore, dMAS predicted nearly five times fewer false positives than the Baseline variant without being negatively impacted by potential false-negative predictions as seen by the KPIs. All dMAS variants had a better performance than the Baseline but made more false-positive predictions than dMAS. A similar pattern is seen for Scenario B, where dMAS scored a precision of 40%. Like in Scenario A, the slight decrease of the KPI values compared to the Baseline has a negligible effect according to the A-tests. Looking at the amount of true positive predictions in Scenario B, we see that the baseline variant had at least 45% more false-negative predictions than dMAS. In Scenario C, the difference between the precision values of dMAS and the Baseline is lower than for the previous two scenarios. dMAS-U(200) was, in fact, the best performing mechanism, while dMAS-U(300) was not able to correctly identify any conflict and, as a result, had the highest $\overline{AC_{tt}}$ of all mechanisms. In Scenario D, the Baseline and dMAS performed very similarly to each other, with the KPIs being slightly in favour of the Baseline, but the difference was negligible. The dMAS variants, on the other hand, performed better, with dMAS-U(400) scoring the highest precision of all.

From the above, we deduce that the precision of dMAS-CBS is substantially higher for scenarios with a higher number of aircraft movements and greater coupled traffic. During runway reconfigurations, the lower traffic levels and coupling between aircraft make dMAS base predictions more prone to Type I errors. The undersampling variants, on the other hand, have shown good performance during runway reconfiguration, except from dMAS-U(300), which failed to positively identify future conflicts.

7 Discussion

This study demonstrated that machine learning could improve the conflict prediction performance of a CBS-based cooperative coordination and planning mechanism. Four dMAS based mechanisms were tested, of which three were trained on under-sampled datasets.

The pure dMAS mechanism offered more precise predictions and matched the overall system performance of the Baseline mechanism. In addition, its conflict prediction precision was found to be significantly better in scenarios at which the airport experiences capacity peaks. However, the extent to which undersampling improves our proposed learning-based approach is not clear. More research on this topic needs to be conducted. The dMAS-U(200) and dMAS-U(400) variants seem to underperform in traffic conditions with a high number of movements by allowing bottlenecks to be created and preventing the smooth flow of traffic. During runway reconfigurations, the dMAS variants, except from dMAS-U(300), were shown to offer a higher conflict detection precision than the Baseline and pure dMAS variants. When looking at the CBS_{AE}, however, improvements only in one of the two considered scenarios were observed. The pure dMAS, on the other hand, was found to be more consistent in terms of both prediction error and precision than the dMAS variants and was able to operate successfully to the full extent of the provided flight schedule. Undersampling reduces the training data size to allow the ratio of rare to normal samples to be greater. In certain cases, this might lead to insufficient training, especially if the number of rare samples is small. As a result, the mechanisms may be more prone to incorrect predictions. We recommend the use of other sampling techniques to bias the learning algorithms, such as oversampling or a combination of both [34].

Concerning the decision around the learning task, a few points can be made. Firstly, the choice of the learning model to predict link specific traversal times was purely based on the choices made in a previous work [12]. Other ML algorithms such as SVMs [44], or Random Forests [45], which have been used in the past in the taxi-time estimation literature [11], could offer better estimates than ANNs. A more elaborate model selection process is recommended for future research. Secondly, the ML models are trained offline using data that are generated during simulation runtime. An interesting direction to investigate would involve implementing online learning in the system, allowing the models to adapt their predictions according to the changing circumstances at the airport. The concept was trialled in the early stages of this study, but issues were found during the online feature normalisation caused by the in-variance of feature values at specific simulation periods. Tree-based models [46] which do not require data normalisation would be better candidates to implement.

In this study, only a limited set of data was available, and therefore semi-random flight schedules were used as inputs in the data generation process. Although the schedules produced by the flight schedule generation mechanism resemble reality to some extend, not all real-life traffic patterns are captured. Models trained on actual data instead are expected to have greater generalisation abilities than the current case. Moreover, it would allow the assessment of the proposed mechanisms on a broader set of operational scenarios.

In addition, a sensitivity analysis on the model parameters was not performed in this study. It was deemed infeasible given the number of parameters and the computational complexity involved in the process. It would be

of interest to determine the response of the MAS model in variations of the aircraft, CBS and dMAS parameters. This would allow generalising better the conclusions found. Furthermore, we did not differentiate between input features per type of taxiway link in this study, and all ML links were allocated the same features. However, depending on their location and size, links are subject to different traffic patterns, which other combinations of features can capture. Therefore, we recommend analysing the feature importance on links of similar properties in order to identify better fitting features.

From an operational perspective, simulation experiments were performed using fixed aircraft kinematic parameters such as maximum and minimum acceleration levels and speeds. The parameters were chosen based on the A320 and B737 aircraft types, which are largely operated at AAS. In reality, however, various aircraft kinematic profiles are present at AAS. The MAS model can therefore be extended to incorporate variable aircraft performances. Future work should also consider incorporating the apron operations into the model and the effects of the flight crew behaviour. Regarding the latter, in real life, crews do not always taxi at the maximum taxi speed in order to reduce fuel consumption or to fulfil safety requirements. Furthermore, certain aspects in their behaviour, such as uncertainty or unfamiliarity with the taxiway layout, may prevent them from timely executing their allocated plans. These additions would increase the simulations' realism and allow for a more consolidated assessment of the implemented mechanisms.

8 Conclusion

This study has taken the first step to investigate how machine learning can be used to enhance the performance of a cooperative multi-agent path planning algorithm in the context of airport surface movement operations. The proposed learning-based mechanism aims to replace a deterministic conflict detection algorithm that is based on the propagation of the instantaneous aircraft dynamics.

The proposed mechanism consists of a delegated Multi-Agent System (dMAS) that runs on top of a distributed version of the Conflict Based Search (CBS) Multi-Agent Path Finding (MAPF) algorithm and a set of single-layered Artificial Neural Networks (ANNs), each responsible for modelling the traversal time dynamics of specific taxiway segments. The dMAS uses the notion of intention propagation where ATC Agents inform other ATC Agents about changes in the intentions of aircraft under their control. Information propagation is performed by means of lightweight agents, called Mobile Agents. Each Mobile Agent holds information related to a single Aircraft Agent. When deployed, the Mobile Agent traverses the intended path of the Aircraft Agent and, by querying the ANNs, builds a timeline of the associated estimated times of arrivals. Relevant ATC Agents use this information to update their internal state and the expected future aircraft count on taxiway segments that coincide with their location. Together with other predictors that capture local taxiway dynamics, the updated future intention levels are also used during a CBS prediction cycle. Similarly to the intention propagation process, Mobile Agents are deployed, and instead of notifying ATC Agents, they only return the queried timeline estimates. The CBS agent who maintains the conflict prediction logic compares the estimated timelines for potential future conflicts and notifies the relevant ATC Agent if needed.

At the beginning of our analysis, all ANNs were trained on data generated by dMAS during runs on semi-random flight schedules. In addition to the dMAS-CBS mechanism, we also studied the effect of biasing the ANNs to give greater importance to samples that corresponded to longer taxi times. The technique of undersampling was used for that purpose, and three additional dMAS variants with undersampling ratios of 200, 300 and 400 were created. We then performed simulation experiments using a real-world flight schedule.

Concerning dMAS-CBS, it was found that although it was able to control aircraft traffic safely, the effect on the overall system behaviour compared to the original approach was insignificant. A closer analysis performed on the conflict prediction level showed that dMAS-CBS is more conservative than the Baseline-CBS when making predictions. In almost all of the tested operational scenarios, dMAS-CBS handled fewer conflicts than its counterpart without negatively affecting the system dynamics. The measured error distributions indicate less variance in the dMAS-CBS predictions. When looking at the precision of the mechanisms, it was found that dMAS-CBS is, on average, more precise. In a scenario where the airport experienced arrival peaks, the precision of dMAS-CBS was measured to be 51% compared to 22% of the baseline mechanism. More specifically, dMAS-CBS predicted five times fewer false positives than the baseline. During departure peaks, a precision of 40% was measured for dMAS-CBS while only 23% for the baseline. During periods with a high number of runway reconfigurations, dMAS had a similar performance to the baseline, which can be explained by the absence of actual conflicts in the datasets, as the airport operates at a reduced capacity.

Regarding the undersampling variants, it was found that their performance in scenarios of peak traffic was lower than dMAS-CBS, both in terms of conflict prediction error and precision. In certain cases, they could not safely control traffic which resulted in the formation of bottlenecks. In one of the scenarios of multiple reconfigurations, the variants offered better precision than dMAS. However, we cannot conclude whether these are more suited for such scenarios, as their conflict prediction error was found to be higher than dMAS-CBS in another similar scenario. Other ways of biasing the learning algorithm should therefore be investigated in

future research.

References

[1] EUROCONTROL, "Eurocontrol five-year forecast 2020-2024," 2020.

[2] EUROCONTROL, "European aviation in 2040, challenges of growth, annex 1, flight forecast to 2040," 2018.

[3] H. Udluft, *Decentralization in Air Transportation.* Phd thesis, Faculty of Aerospace Engineering - Delft University of Technology, 2017.

[4] T. Noortman, *Agent-Based Modelling of an Airport's Ground Surface Movement Operation.* Master's thesis, Faculty of Aerospace Engineering - Delft University of Technology, 2018.

[5] K. Fines, A. Sharpanskykh, and M. Vert, "Agent-based distributed planning and coordination for resilient airport surface movement operations," *Aerospace*, vol. 7, no. 4, p. 48, 2020.

[6] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[7] H. Idris, J.-P. Clarke, R. Bhuva, and L. Kang, "Queuing Model for Taxi-Out Time Estimation," *Air Traffic Control Quarterly*, vol. 10, no. 1, pp. 1–22, 2002.

[8] P. Balakrishna, R. Ganesan, and L. Sherry, "Application of reinforcement learning algorithms for predicting taxi-out times," *Proceedings of the 8th USA/Europe Air Traffic Management Research and Development Seminar, ATM 2009*, pp. 255–261, 2009.

[9] S. Ravizza, J. A. Atkin, M. H. Maathuis, and E. K. Burke, "A combined statistical approach and ground movement model for improving taxi time estimations at airports," *Journal of the Operational Research Society*, vol. 64, no. 9, pp. 1347–1360, 2013.

[10] J. Chen, S. Ravizza, J. A. Atkin, and P. Stewart, "On the utilisation of fuzzy rule-based systems for taxi time estimations at Airports," *OpenAccess Series in Informatics*, vol. 20, pp. 134–145, 2011.

[11] H. Lee, W. Malik, and Y. C. Jung, "Taxi-out time prediction for departures at charlotte airport using machine learning techniques," *16th AIAA Aviation Technology, Integration, and Operations Conference*, no. June, pp. 1–11, 2016.

[12] R. Claes, *Anticipatory Vehicle Routing: Coordinating traffic using community generated traffic predictions.* Phd thesis, Faculty of Engineering Science - Katholieke Universiteit Leuven, 2015.

[13] T. Holvoet and P. Valckenaers, "Exploiting the environment for coordinating agent intentions," in *International Workshop on Environments for Multi-Agent Systems*, pp. 51–66, Springer, 2006.

[14] T. Holvoet, D. Weyns, and P. Valckenaers, "Patterns of delegate mas," in *2009 Third IEEE international conference on self-adaptive and self-organizing systems*, pp. 1–9, IEEE, 2009.

[15] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[16] S. Polydorou, *A learning based approach for distributed airport surface movement operations.* Master's thesis, Faculty of Aerospace Engineering - Delft University of Technology, 2021.

[17] M. H. C. Torres and T. Holvoet, "Self-adaptive resilient service composition," in *2014 International Conference on Cloud and Autonomic Computing*, pp. 141–150, IEEE, 2014.

[18] M. Ferber, T. Rauber, M. H. C. Torres, and T. Holvoet, "Resource allocation for cloud-assisted mobile applications," in *2012 IEEE Fifth International Conference on Cloud Computing*, pp. 400–407, IEEE, 2012.

[19] R. Claes and T. Holvoet, "Ad hoc link traversal time prediction," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 1803–1808, IEEE, 2011.

[20] B. Irie and S. Miyake, "Capabilities of three-layered perceptrons.," in *ICNN*, pp. 641–648, 1988.

[21] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural networks*, vol. 2, no. 3, pp. 183–192, 1989.

[22] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[24] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*, pp. 65–93, Elsevier, 1992.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization.," *Journal of machine learning research*, vol. 13, no. 2, 2012.

[27] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *arXiv preprint arXiv:1206.2944*, 2012.

[28] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[29] J. H. Friedman, "Stochastic gradient boosting," *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[30] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of bayesian methods for seeking the extremum," *Towards global optimization*, vol. 2, no. 117-129, p. 2, 1978.

[31] S. Markov, "Skopt documentation," 2017.

[32] P. Branco, L. Torgo, and R. Ribeiro, "A survey of predictive modelling under imbalanced distributions," *arXiv preprint arXiv:1505.01658*, 2015.

[33] M. Kubat, S. Matwin, *et al.*, "Addressing the curse of imbalanced training sets: one-sided selection," in *Icml*, vol. 97, pp. 179–186, Citeseer, 1997.

[34] L. Torgo, P. Branco, R. P. Ribeiro, and B. Pfahringer, "Resampling strategies for regression," *Expert Systems*, vol. 32, no. 3, pp. 465–476, 2015.

[35] P. Branco, L. Torgo, and R. P. Ribeiro, "Smogn: a pre-processing approach for imbalanced regression," in *First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, pp. 36–50, PMLR, 2017.

[36] L. Torgo and R. P. Ribeiro, "Utility-based regression," in *European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 597–604, Springer, 2007.

[37] R. P. A. Ribeiro, *Utility Based Regression*. Phd thesis, Dep. Computer Science, Faculty of Sciences - University of Porto,, 2011.

[38] L. Torgo and R. Ribeiro, "Precision and recall for regression," in *International Conference on Discovery Science*, pp. 332–346, Springer, 2009.

[39] P. Branco, *Re-sampling approaches for regression tasks under imbalanced domains.* Master's thesis, Dep. Computer Science, Faculty of Sciences - University of Porto,, 2014.

[40] C. Van Rijsbergen, *Information Retrieval*. Dept. of Computer Science, University of Glasgow, 2 ed., 1979.

[41] F. Klügl, "A validation methodology for agent-based simulations," in *Proceedings of the 2008 ACM symposium on Applied computing*, pp. 39–43, 2008.

[42] R. Sargent, "Verification and validation of simulation models," vol. 37, pp. 166 – 183, 01 2011.

[43] A. Vargha and H. D. Delaney, "A critique and improvement of the cl common language effect size statistics of mcgraw and wong," *Journal of Educational and Behavioral Statistics*, vol. 25, no. 2, pp. 101–132, 2000.

[44] Y. Tian, Y. Shi, and X. Liu, "Recent advances on support vector machines research," *Technological and Economic Development of Economy*, vol. 18, 03 2012.

[45] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[46] G. James, D. Witten, T. Hastie, and R. Tibshirani, "Tree-based methods," in *An introduction to statistical learning*, pp. 303–335, Springer, 2013.

II

Literature Study previously graded under AE4020

1

Introduction

Air traffic has been experiencing significant growth over the last decades. In Europe alone, the number of Instrument Flight Rules (IFR) flights is expected to reach a total of 16.2 million by 2040, a 53% increase from the flight year of 2017 [31]. This increasing trend has been creating bottlenecks in the Air Traffic Management (ATM) system which is limited in terms of capacity and resources. To cope with such challenges, initiatives such as the Single European Sky (SESAR) in Europe and Next Generation Air Transportation System (NextGen) in the United States have been established in order to modernize the current Air Traffic Control (ATC) system while meeting the appropriate safety and environmental standards. One of key areas at focus are airports who are among the key players to contribute to these bottlenecks. In the third quarter of 2019, 9% of the total delay encountered was caused by airport operations. Issues such as arrival and departure management, gate assignment and surface movements operations have thus received a significant attention by the research community lately.

In this thesis, the focus lies on the airport surface movement operations problem which is best described as a path planning problem. The goal is to find conflict free routes for aircraft through the taxiways to their destination as quickly as possible, while reducing the overall taxi time and/or other objectives such as taxi distance. This is a challenging task as factors such as size and airport layout, the number of aircraft present and weather conditions have a direct influence on the complexity of the problem. In addition, one needs to consider the various dependencies between taxiing aircraft [2]. Currently, movement operations are handled by Air Traffic Controllers (ATCOs) in a way which resembles operations of a centralized system. ATCOs are limited in terms of mental capacity and their workload and performance can vary depending on the conditions existing on the airport's surface.

Research performed in the Air Transport and Operations section of TU Delft has proposed a concept, using the Agent-Based Modelling (ABM) methodology, in which movement operations are handled in a distributed manner. Agent based modeling (ABM) is a technique which allows the modeling of interactions between agents in complex socio-technical systems and provides the user with the means to identify emergent behaviors and system wide effects which would not have been the case with other modeling techniques [76]. Particular attention has been spent to the ground operation of Amsterdam Airport Schiphol (AAS), one of the busiest and most complex airports in Europe. An existing model realizes distributed control by allocating tasks to virtual controller agents placed on the taxiway infrastructure. Each controller is responsible for handling traffic in its local environment as well as coordinate and plan activities with agents in its vicinity. It has been shown that such a distributed control system is able to match the performance of the ATCOs under nominal system states and demonstrate a good behavior when runway reconfigurations take place.

This MSc thesis continues in the same research direction. The focus is to enhance the current implementation and make it more suitable for applications in real-world like scenarios. Two ways which are thought to enhance the current system have been identified in this literature study. The first involves the introduction of a learning mechanism based on which the agents will perform their tasks in a more efficient manner, and the second involves the modification of the existing planning mechanism to a variant which is shown to be more efficient in terms of scalability and runtime.

This literature study is structured in the following way. First, in Chapter 2, the airport surface movement operations as currently being performed at AAS are discussed. The focus in this chapter is to obtain an understanding of the processes and interactions between the stakeholders responsible for managing aircraft

movements as well as to describe some of the challenges they face. Following this introduction, Chapter 3, outlines the main features of previous research performed in this department, on the domain of airport surface movement operations, resulting in an identification of a research gap which this thesis will aim to fill. Chapter 4, discusses multi-agent learning, one of the aspects thought to address the identified research gap. At first, opportunities in which the existing distributed control system can be enhanced through a learning mechanism are identified and the most relevant is chosen. Subsequently, existing literature which cover certain aspects of the identified learning task is reviewed, and a course of action is formulated to some extent. The key for the implementation of efficient distributed control is to have agents plan their actions. An existing implementation of a multi-agent planning mechanism has shown promising results; however, new methods proposed in the literature are still worth to be investigated. For this reason, in Chapter 5 a number of multi-agent path planning (MAPP) algorithms covered in the literature are presented and a trade-off is performed in order to select the most suitable candidate. Lastly, the findings from this literature study are combined in Chapter 6 to a research proposal, in which a main research question and several sub-questions are derived.

2

Airport Surface Movement Operations at Schiphol Airport

The focus in this chapter is to present information on how airport surface movement operations are being performed within a complex airport. The airport at focus is Amsterdam Airport Schiphol (AAS) which has also been extensively studied in [36, 60]. Schiphol is one of Europe's largest airport consisting of 6 runways and a large taxiway network. In 2018 it was connected to 327 destinations worldwide by 108 airlines and accommodated a total of 71.1 million passengers, making it the 11th busiest airport in the world [1, 70]. It is also the hub airport for KLM, KLM Cityhopper, Martinair, Transavia and TUI fly Netherlands. The chapter is divided as follows. First, in Section 2.1, a description of the Schiphol Collaborative Decision Management system is given. In Section 2.2, a summary of the surface movement operations is given while Section 2.3 describes the socio-technical system of Schiphol, identifying all agents and their interactions. Section 2.4 deals with the operational aspect of runways re-configurations and lastly, in Section 2.5, uncertainty factors which play an important role in surface movement operations are identified.

2.1. Schiphol Collaborative Decision Management

The information presented in this section has been derived from Eurocontrol's Airport CDM Implementation Manual [30] and Schiphol's CDM Operations Manual [71].

Ground operations at Schiphol are performed according to the Airport Collaborative Decision Making system (A-CDM) introduced by EUROCONTROL. According to the Eurocontrol Airport CDM Implementation manual, the system aims at improving the Air Traffic Flow and Capacity Management (ATFCM) at airports by reducing delay, improving the predictability of events and optimizing the use of the available resources. This is achieved through the timely information exchange between the A-CDM stakeholders [30]. For Schiphol airport these stakeholders are the airlines, ground handlers, Luchtverkeersleiding Nederland[1] (LVNL), and AAS itself. Schiphol-CDM is in line with Eurocontrol's Implementation Manual and aims at optimizing the turnaround process at its entirety. A-CDM consists of 5 processes. These are shown in Figure 2.1. The processes shown in dark blue are elaborated in this section as there are more relevant to consider for this study.

Figure 2.1: A-CDM processes

2.1.1. The Milestone Approach

An important feature of the A-CDM is the Milestone Approach Element. It describes the most significant events that occur for a flight, from its initial planning phase to its take-off. The time at which these events

[1] The Dutch Air Navigation Service Provider

occur is shared between the stakeholders and enables the realization of a common situational awareness and predictability of imminent events. The events are denoted as milestones. The completion of a particular milestone triggers the decision making process for later events and influences the progress of the flight as well as the accuracy at which this progress can be predicted. Figure 2.2 shows the milestones recommended by Eurocontrol.

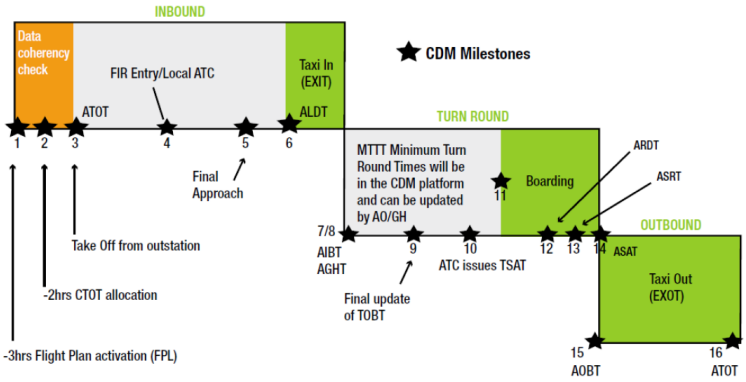

Figure 2.2: Recommended milestones as described in the Eurocontrol Airport CDM Implementation Manual. Retrieved from [30].

The milestones relevant for this study are milestone 6 until milestone 16 (Actual Take-off Time, ATOT). At Schiphol's CDM system, milestones 12 and 14 are not considered. The actions taking place during each milestone are explained in the following:

- **Milestone 6, Aircraft landed:** The Actual Landing Time (ALDT) and the Estimated In-Block Time (EIBT) are updated. The flight state in the system is updated to Flight Taxiing.

- **Milestone 7/8, Aircraft In-Block:** The Actual In-Block Time (AIBT) is updated and the flight state is updated to Flight In-Blocks.

- **Milestone 9, Final Target-Off Block Time (TOBT) update:** TOBT is the time at which the Main Ground Handler (MGH) estimates to have completed all ground handling activities including, the aircraft's doors closed and the boarding bridge and any ground equipment removed. The MGH, who is appointed by the airline operator, is responsible for updating the TOBT.

- **Milestone 10, ATC issues Target Start-Up Approval Time (TSAT):** The TSAT window is the estimated time at which the aircraft is able to start the engines and begin its taxiing. It has a range between -5 and +5 minutes and is computed by the outbound planner using the Collaborative Pre-Departure Sequence Planning System (CPDSP), using the TOBT as an input. More information regarding the CPDSP are given in the following subsection.

- **Milestone 11, Boarding Starts:** The flight state in the A-CDM system is updated by the gate agent to Boarding.

- **Milestone 13, Actual Start-Up Request Time (ASRT):** ATC sets the ASRT when the pilot calls a ready clearance within the flight's TSAT window. The flight state is then updated to Ready by the system. The Target Take-Off Time (TTOT) is also updated. In the case that the pilot calls ready prior to the assigned TSAT window, the pilot is asked to call ready when the flight is on its TSAT window.

- **Milestone 15, Off-Block:** The Actual Off-Block Time (AOBT) is updated and the flight state is updated to Flight Taxiing.

- **Milestone 16, Take-Off:** The Actual Take-Off Time (ATOT) is updated and the flight state is update to Flight airborne.

2.1.2. Pre-departure Sequence

The Pre-Departure sequence refers to the order at which the aircraft are planned to depart from their stands, taking into account operators preferences and operational constraints. At Schiphol airport, the CPDSP system has been implemented for this purpose. Figure 2.3 shows a diagram of how CPDSP works. By adding the Estimated Taxi-Out Time (EXOT) to the TOBT an earliest estimate of the Target Take-off Time (TTOT) is made. This estimate is then fed to an optimization algorithm together with information regarding the current runway usage and required separation times/distances based on the flight's Standard Instrument Departure (SID) and Wake Turbulence Category (WTC). The output of the algorithm is an optimized TTOT which when subtracted by the EXOT leads to the optimized TSAT.

The order at which the CPDSP plans the flights is the following. Flights which are constrained by the Network Manager Operations Centre (NMOC) of Eurocontrol have a higher priority. These flights are given an Air Traffic Flow Management (ATFM) departure slot which restricts their departure to a window between 5 minutes before and 10 minutes after the Calculated Take-Off Time (CTOT). These departure slots are issued when the NMOC foresees an imbalance between demand and capacity at airports or along a flight's route [32].

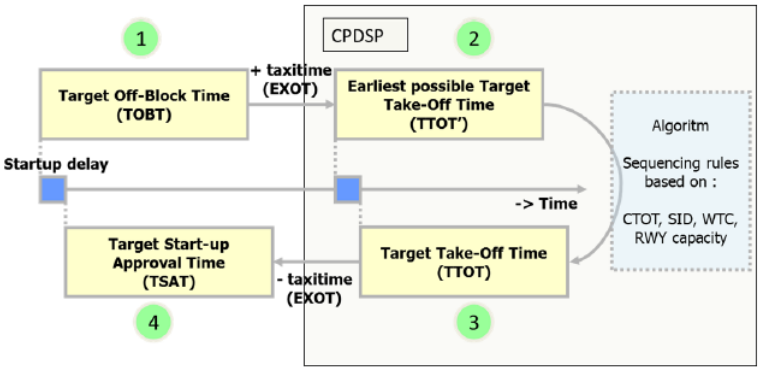

Figure 2.3: Collaborative Pre-Departure Sequence Planning. Retrieved from [71].

2.1.3. Collaborative Management of Flight Updates

According to Eurocontrol's definition, collaborative management of flight updates requires the exchange of flight update messages between the Network Manager (NM) and the A-CDM airport, the provision of landing time estimates and lastly the improvement of the ATFM slot management process for departing flights. At Schiphol, the Eurocontrol Flight Update Messages information is used by the ATC to determine Estimated Landing Times (ELDT) for inbound flights even before these flights are assumed by the ATC. Furthermore, Schiphol-CDM is linked to NMOC via a departure planning information connection. This connection enables local stakeholders to reduce ATFM delays, optimize their runway capacity and turnaround processes. It is also beneficial to the whole ATM network (ANSPs and AOs) as it enables the generation of more accurate flight profile estimates and informs when a flight is canceled.

2.2. Summary of surface movement operations at Schiphol

Having explained the main components of Shicphol's-CDM system, a summary of surface movement operations is given in this section. The information has been derived from two previous MSc studies [36, 60]. Figure 2.4 depicts a series of surface movement operations phases that are typical at any airport. For convenience it is used to describe the operations as done in Schiphol.

In the first phase, a flight lands after receiving a landing clearance from the runway controller. The ALDT is recorded right after touchdown and shared through the A-CDM portal. The aircraft is required to vacate the runway as soon as it reaches a certain speed, so that the next aircraft in the arrival sequence can safely land. Aircraft can exit the runway with a higher speed via a rapid exit taxiway. These are taxiways with intersection angle of 30° to the runway. The maximum speed at which aircraft can exit the runway via a rapid exit is set by ICAO to be 35kts [42], although in some cases exits of up to 50kts have been observed, depending on the radius of the taxiway at turn-off.

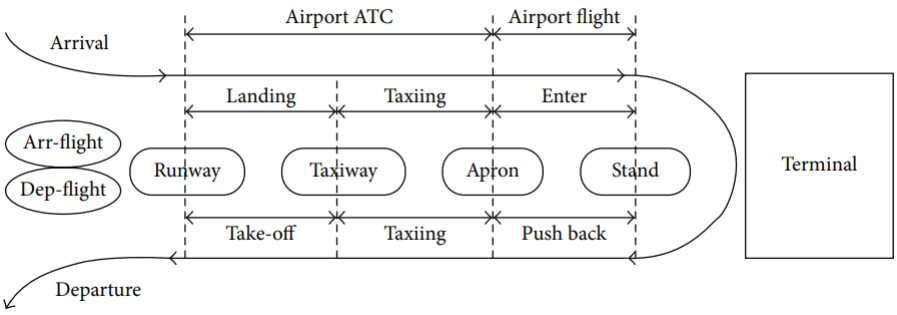

Figure 2.4: Ground operations flow in an airport. Retrieved from [47].

After the aircraft has vacated the runway, it is handed over to the ground controller who is responsible for providing taxi route information to the assigned aircraft stand. The route provided to the aircraft is used to estimate EXOT which is then recorded to the CDM system. In the case that the stand at which the aircraft must taxi is not available, due to for example delays of the departing aircraft, the ground controller might issue new taxi instructions to an empty stand or instruction to hold inside or outside the apron until the stand becomes available again. If the waiting time is expected to be long, the aircraft might be instructed to taxi towards a remote holding area in order not to block other traffic.

Once the arriving aircraft makes it to the stand, the engines are shut down and the ground handler starts with the turnaround activities. These include, passenger boarding/disembarking and cargo loading/unloading, cleaning, maintenance checks and refuelling. At the same time the AIBT is recorded at Schiphol's CDM. The aircraft is then prepared for its next flight.

During the turnaround process, the MGH issues a TOBT which the delivery controller uses to compute a TSAT. Once all activities have been completed, the flight crew communicates a ready clearance at time ASRT to the delivery controller. The crew then contacts the ground controller and requests a push back and engine start-up clearance. Following specific stand/apron procedures, the controller gives the respective clearance after checking the nearby traffic and whether the flight has called ready within its assigned TSAT window.

At the timepoint at which the aircaft starts its push-back, the AOBT is recorded to the CDM system. When the crew is ready to taxi, they contact the ground controller who then provides them with taxiing instructions towards the holding point of the runway used for departures. Once again, EXOT is estimated and recorded in the CDM system. At the holding point, the responsibility of the flight is passed on to the tower controller who then gives a take-off clearance at ATOT.

2.3. Schiphol socio-technical system

In the previous sections it was seen that the ground operations at Schiphol are performed concurrently via a number of different agents. Figure 2.5 presents these agents and the agents with whom they interact. The figure is further elaborated in subsection 2.3.1 where the goals and responsibilities of each agent are explained and in subsection 2.3.2 where the interactions between on another are identified. The information presented here has been derived from two previous Msc studies [36, 60].

2.3.1. Agent types

The socio-technical system of ground operations at Schiphol consists of 6 agents. These are:

Aircraft

Aircraft are controlled by their flight crews. They have properties such as: wingspan, weight, number of engines and WTC. The flight crews are expected to follow the commands given by the ATC and act in accordance with the rules of air defined by ICAO[2]. In the context of surface movement operations, aircraft are distinguished between arrival and departing aircraft. The goal of the arriving aircraft is to travel from the runway to their assigned gate in an optimal manner with respect to fuel burn and/or taxi time. For departing aircraft their goal is to reach the runway from their gates in an optimal manner.

[2]Annex 2: Rules of Air [43]

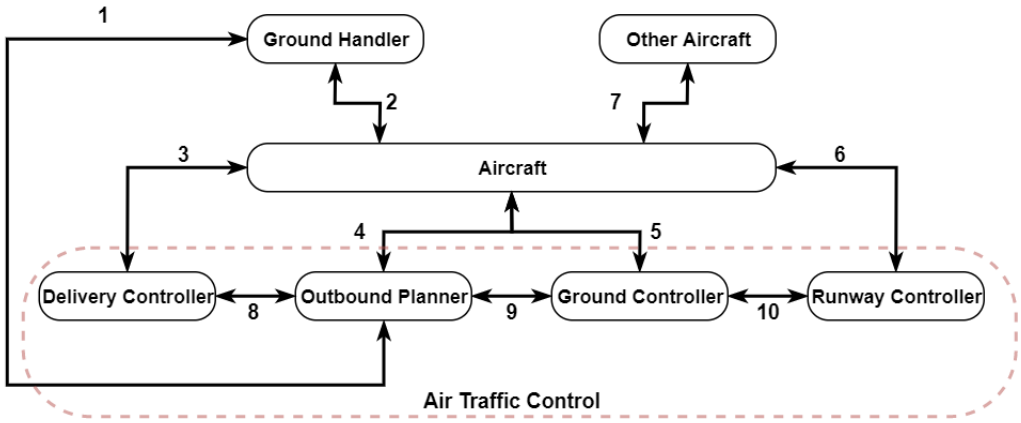

Figure 2.5: Diagram of the socio-technical system responsible for the ground operations at Schiphol airport. Retrieved from [60].

Ground Handler

The ground handler is responsible for performing all pre-flight and post-flight operations on the aircraft while that is parked at the stand. These include: passenger boarding/disembarking and cargo loading/unloading, cleaning, maintenance checks and refuelling. In addition, the ground handler is responsible for estimating a TOBT in the A-CDM system which the delivery controller uses then to compute a TSAT. The goal of the ground handler is to minimize the turnaround time of the aircraft and ensure that the difference between the AOBT and the TOBT entered in the system is kept to a minimum.

Delivery Controller

The delivery controller is responsible for issuing route clearances to the departing aircraft. This is done by verifying their respective flight plans, issuing a SID with respect to the departing runway and a squawk code such that the aircraft can be identified. If an aircraft is assigned a CTOT by NMOC this is communicated via the delivery controller. Furthermore, a clearance request received by the aircraft's flight crew which is assigned a CTOT, is checked to be within the CTOT window prior to issuing the clearance. The goal of the delivery controller is to verify flight plans, provide each departing aircraft a SID and a squawk code and receive a confirmation by the flight crew about the issued instructions. Depending on the workload, the task of the delivery controller and the outbound planner can be performed by the same controller.

Outbound Planner

After the flight crew has reported that the flight is ready to start, the outbound planner determines whether the flight can actually be given a start-up clearance. The decision depends on the congestion levels of the airport at the given time. The outbound planner makes use of the CPDSP tool to determine TSAT sequences for the flights such that the formation of queues at the runways is avoided. The tool uses as input the earliest estimate of target take-off time (TTOT') which is equal to the TOBT plus the estimated taxi time (EXOT). If there are many aircraft with a similar TTOT' then queues will from at the runways. Based on sequencing rules such as CTOT, SID, WTC and runway capacity (check Figure 2.3), a sequencing delay is added to each flight making their TSAT and hence their TTOT different. The goal of the outbound planner is to regulate the flow of traffic based on the TSATs issued by the CPDSP tool and grant start-up clearances.

Ground Controller

The ground controller is responsible for managing all airport surface movement operations. This includes the granting of push-back clearances and taxi instructions for both arriving and departing aircraft, as well as the control of ground vehicles. The ground controller is actively monitoring the apron and taxiway system and prevents collisions from happening. The ground controller is also responsible for making sure that the departing aircraft timely reaches the runway and is sequenced in accordance to the CPDSP tool. The goal of the ground controller is to safely and efficiently guide the aircraft and ground vehicles to their destination.

Runway Controller

The runway controller is responsible for managing all operations related to the use of the runway system. This includes: granting take-off and landing clearances based on the departure sequence, ensuring aircraft separation based on WTC, granting runway crossing permissions as well as controlling ground vehicles which perform runway inspections. The goal of the runway controller is to ensure that the above mentioned activities are executed as safely and efficiently as possible.

2.3.2. Interaction between agents

The interactions between the agents as shown in Figure 2.5 are described below:

1. **Ground Handler & Outbound Planner:** Based on the progress of the turnaround processes performed on an aircraft, the ground controller estimates the TOBT which is passed to the outbound planner through the A-CDM portal.

2. **Ground Handler & Aircraft:** The ground handler informs the aircraft's flight crew regarding the progress of the turnaround processes. The flight crew informs the ground handler of any issues related to the TOBT/TSAT issued by the outbound planner. In addition, the flight crew can make requests to the ground handler for additional services.

3. **Aircraft & Delivery Controller:** The flight crew requests a flight plan clearance from the delivery controller either through radio communication or Aircraft Communications and Reporting System (ACARS) or Controller-Pilot Data Link Communications (CPDLC). The flight crew also acknowledges clearances and instructions given by the controller. The delivery controller grants delivery related clearances to the aircraft's flight crew. This includes: SID, squawk code and other flight plan related clearances. The delivery controller must receive clearance and instruction confirmations by the flight crew. The controller informs the flight crew to contact the outbound planner.

4. **Aircraft & Outbound Planner:** The flight crew notifies the outbound planner that they are ready to start-up. By that time all turnaround related activities must have been completed. The outbound planner grants permission to the crew to start-up based on whether the request made is within the designated TSAT window. The planner communicates information like the Automatic Terminal Information Service and barometric pressure level (QNH) to the flight crew and requests them to contact the ground controller.

5. **Aircraft & Ground Controller:** The flight crew states the gate at which their aircraft is standing and requests a pushback clearance. Once pushback is completed, they notify the ground controller that the aircraft is ready to taxi. The ground controller grants pushback permission to the flight crew when this is permissible as well as taxi instructions after the pushback is completed. During taxiing, the controller monitors the ground traffic and advises the flight crew accordingly. Once close to the runway holding point, the ground controller asks the flight crew to contact the runway controller.

6. **Aircraft & Runway Controller:** The flight crew notifies the runway controller that they are ready for departure and they read back any clearances issued by the controller. The runway controller communicates instructions related to holding short, taking-off, landing and runway crossing to the flight crew. The controller also notifies arrival and crossing aircraft to contact the ground controller.

7. **Aircraft & Other Aircraft:** The flight crew observes nearby traffic and is responsible for maintaining visual separation. In addition, the crew must comply with ICAO's rules of air [43].

8. **Delivery Controller & Outbound Planner:** The delivery controller transfers the responsibility of a flight to the outbound planner, once the clearance is given and read back by the flight crew. The handover is made through the use of Electronic Flight Strips (EFS). The outbound planner hands over the responsibility of the flight in the same manner as before when changes to the clearances are required. For example, in the scenario of a sudden runway reconfiguration a new SID has to be given to the departing aircraft.

9. **Outbound Planner & Ground Controller:** The outbound planner transfers the responsibility of a flight to the ground controller, once the start-up clearance has been given. The responsibility of the flight can be handed back to the outbound planner if an aircraft experiences an issue which does not allow it to pushback or taxi. The handover in both cases is performed by means of the EFS.

10. **Ground Controller & Runway Controller:** When an aircraft is approaching the runway holding point or needs to cross an active runway, the responsibility is handed over to the runway controller. After an aircraft has vacated the runway either after a landing or a runway crossing, the responsibility is handed to the ground controller. The handover in both cases is performed by means of the EFS.

2.4. Runway configurations

An important operational aspect of the surface operations at Schiphol airport is the frequent occurrence of runway re-configurations. On average, runways at Schiphol are reconfigured 14 times during the day [60]. When a reconfiguration takes place, the active landing and departing runways changes and the airport enters a transient state of operations. During such states the ATC needs to determine new routes for the arriving and departing aircraft while taking into account the routes of the aircraft taxiing at the moment of the reconfiguration. The workload of the ATCO increases which might result in a temporary decrease in the airport's capacity. Subsection 2.4.1 describes the types of configurations which are commonly seen at Schiphol, followed by a discussion regarding the factors that contribute to runway reconfigurations in subsection 2.4.2. Lastly, in subsection 2.4.3 insights as to how the reconfigurations are planned and executed are provided.

2.4.1. Runway configurations at Schiphol

Not all runways at Schiphol are used for take-offs or landings. A distinction between runways used for arrivals and departures is made in Table 2.1 [57].

Runway	Arrivals	Departures
36R	✓	✗
36C	✓	✓
36L	✗	✓
18L	✗	✓
18C	✓	✓
18R	✓	✗
09	✓	✓
27	✓	✓
22	✓	✓
04	✓	✓
06	✓	✓

Table 2.1: Runway availability per traffic flow.

An analysis performed by Fines [36] using one month data in the winter period of 2018 identified the most common runway configurations, shown in Table 2.2.

Runway Combination	**Time Used [mins]**	**Percentage of Total Time [%]**
18R+24	7835	20.2
06+36L	6170	15.9
18R+24, 18L	5190	13.3
18R, 18C+18L	3505	9.0
06, 36R+36L	2110	5.4
06+36L, 36C	2070	5.3
18R, 18C+24, 18L	2060	5.3

Table 2.2: Commonly used runway configurations at Schiphol based on 1 month of operations during the winter 2018. The configurations are given in (X+Y) format where X are the arrival runways and Y the departing. In case of multiple arrival and/or departing runways a comma (,) is used to separate them. Retrieved from [36].

2.4.2. Factors influencing runway configurations

The main factors influencing the runway configurations at Schiphol are weather, noise abatement procedures and airport demand.

Weather

According to LVNL [58], the runway combination is firstly determined by the weather conditions. Every airport operates under certain wind limits and visibility conditions in order to maintain safe and efficient operations. If aircraft were to depart with tailwind components, for example, larger take-off distances would be required to reach the take-off speeds and therefore the total runway capacity would decrease. From a safety point of view, landing with tailwind or crosswind is also undesirable. There have been cases in the past were aircraft would not have enough runway length to stop or have been blown-off due to severe crosswinds. As a rule, aircraft take-off and land opposite to the wind's direction. According to ICAO [61], typically tailwind components are not allowed to exceed $10kts$ while crosswind components are limited to 15-25kts. Additionally, aircraft manufacturers also place their own limits for the maximum tailwind/crosswind that their aircraft are allowed to experience. Above those limits the aircraft are not safe to perform a landing or take-off. At Schiphol airport the tailwind and crosswind limits are set to $7kts$ and $20kts$ respectively and can change depending on the visibility range and cloud ceiling [60]. The runway conditions which can be either dry or wet can influence these limits.

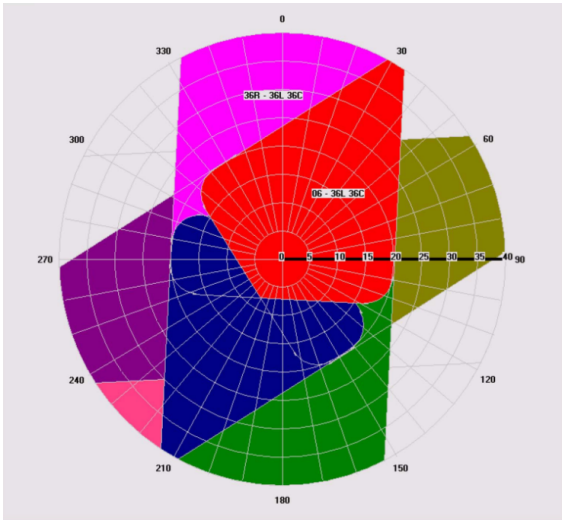

Figure 2.6: Runway availability windrose at Schiphol airport. Retrieved from [66].

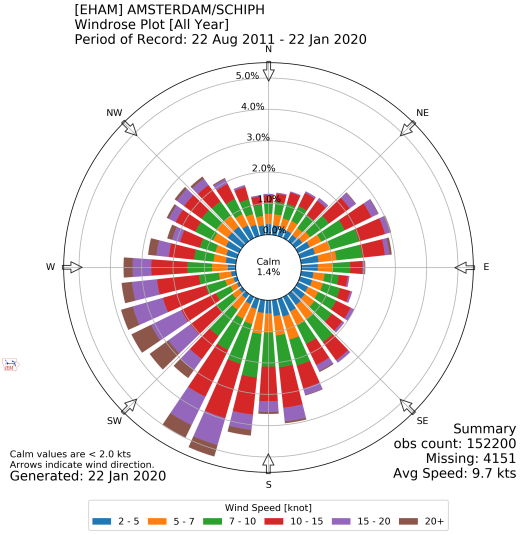

Figure 2.7: Average wind speeds and direction present at Schiphol airport for the period between 08/2011 - 01/2020. Retrieved from [94].

Figure 2.8: Windrose of Schiphol airport.

Figure 2.6 shows the runway availability windrose used by ATCOs to decide on permissible runway con-

figurations. The windrose diagram shows which runways are allowed to be used based on current wind speed and direction. For example, assume that at a given day the airport is experiencing 25kts coming from direction 100°. This means that a possible runway combination during an arrival peak, would be to have runways 18C and 18R for arrivals and runway 18L for departures. At the extreme scenario of having 25kts wind coming from direction 300°, no runway can be utilized as all of them exceed their crosswind and tailwind limit. However, according to Figure 2.7 such a phenomenon is quite rare. The figure shows the wind speed and direction distributions at Schiphol airport for the period between August 2011 and January 2020. It can be seen that South-South West winds are the most dominant, followed by West-South West winds.

Apart from wind, visibility and cloud ceiling also influence the combination of runways which can be used. According to LVNL [58], the poorer the visibility, the greater separation aircraft need. When the visibility is low the runways' landing instrument landing systems (ILS) are used to help with landings. When the visibility is greater than $5km$ and the ceiling is higher than $1000ft$ no special precautions are applied. If the visibility is between $1500m$ and $5km$ and/or the ceiling is between 300 and $1000ft$ then visual approaches are not permitted and specific procedures are applied to dependent runways. Lastly, if the visibility is less than $1500m$ and the ceiling is lower than $300ft$, low visibility procedures are implemented.

In wintry conditions, the ATC adjusts the airport's capacity according to the condition of the runways, taxiways, parking areas and the provision of de-icing facilities [59]. Clearance plans are created in consultation with the ATC and airlines, so that the airport operator can clear the infrastructure from snow and/or ice. Thunderstorms, hail and icing in the clouds can also influence the usages of the runways.

Noise

The choice of runway use also depends on noise abatement procedures present at Schiphol airport. The procedures state that certain runways and runway configurations cannot be used in certain times during the day. More specifically, during daytime (0600-2230) 2+1 runways (landing, take-off or vice versa) can be used while during nightime (2230-0600) 1+1 runways are used. In addition, during nightime RWY 06-24 and 18R-36L are preferred for take-offs or landings as they are located at a greater distance from residential areas around Schiphol [40]. An alternative is to use runways 18C-36C or 09-27. RWY 18L-36R is not used during the night while RWY 04-22 is closed in night [35].

The noise pollution caused by the air traffic is normally described using two indicators, the L_{den} and L_{night}. Both of them describe the annual noise load experienced. L_{den} or day-evening-night level is based on air traffic throughout the entire day while L_{night} focuses only on night traffic (2300-0700). Figure 2.11 shows the average estimated noise contours for the flight year of 2019. The estimated variation of the contours due to changing weather conditions is also shown. These estimates are based on 47 years of historical weather data [22]. Two things can be noted from the figures: in areas closer to the runways the noise level experienced is higher while low noise levels are encountered over populated areas during the night time due to the noise abatement procedures.

Traffic demand

Schiphol experiences periods during the day in which the outbound or inbound traffic levels are high. This is a typical property of hub airports. Up to 100 flights can be managed in these periods. Based on the traffic demand the ATC uses different runway configurations. More specifically, during inbound peaks, Schiphol operates 2 landing runways and 1 take-off runway. Similarly, during outbound peaks, 1 runway is used for landings and 2 for take-offs. In periods where the inbound and outbound peaks overlap, a configuration of 2 runways for each traffic flow is used [58].

Other factors

In addition to the three main factors described earlier a number of other factors can cause reconfigurations. These occur less frequently and are very difficult to predict. Examples are: aborted take-offs, technical failures on runway systems, emergency landings, aircraft crashes, runway cleaning and runway maintenance.

2.4.3. Planning and execution of reconfigurations

Based on the information presented in two previous Msc studies [36, 60], the planning and execution of runway reconfigurations at Schiphol will be explained in this section.

The majority of re-configurations at Schiphol are planned in advance by LVNL. The planning process consists of the tactical planning and operational planning phases [36]. The tactical planning is performed

with three briefings during the day. This is to ensure that the dynamics of flight schedules and weather, the driving forces of reconfigurations, are captured. The first briefing is performed at the evening of each day and determines a runway schedule up until the midday of the following day. Based on the published flight schedule the inbound and outbound peaks are identified and are used to determine the number of runways that are needed to accommodate the traffic during the associated periods. Weather forecasts are then used to determine the runway directions per arrival and/or departing peaks. The choice is also dependent on the noise abatement procedures. The briefing is repeated in the morning and the midday of the following day, in order to determine the runway schedule between midday and evening and evening on-wards, respectively. One hour to 30 minutes prior to the planned reconfiguration, the approach supervisor and the tower supervisor discuss the set of runways to be selected based on a revision of the flight schedules, the meteorological conditions and operational conditions present at the airport at that time. Having selected the runways in question, the two supervisors then define the moment in time where the reconfiguration will take place. The delay caused by the reconfiguration is also calculated. In order to absorb this delay it might be necessary to activate a new runway for the duration in which the re-configuration is in effect or delay pushback clearances in order to reduce queues from forming at runway holding points. The time of the re-configuration is communicated to the tower controllers in order to update their strategies and is subsequently entered in the CPDSP system in order to generate new clearances and TSAT windows [60].

Figure 2.9: L_{den} average noise contour.

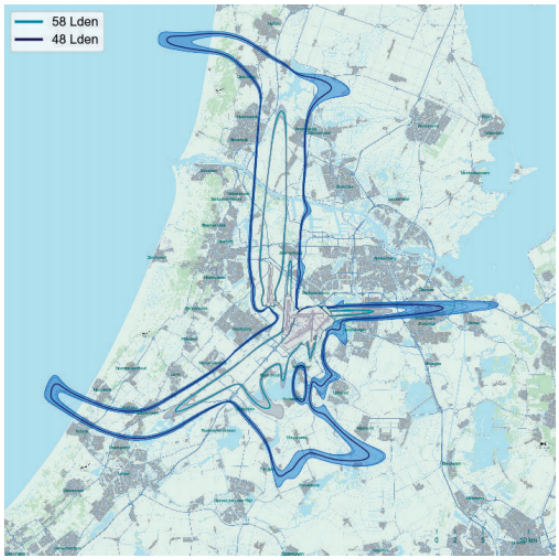

Figure 2.10: L_{night} average contour.

Figure 2.11: Expected average noise level for the flight year of 2019. Retrieved from [22].

2.5. Uncertainty factors in surface movement operations

According to Rappaport et al.[63] the air traffic management systems is a non-linear, highly complex system with many interrelated components and bottlenecks appearing in the system often result from reduced airports' capacity. The authors identified 18 sources of uncertainty and performed quantitative analysis in 4 of those. The identified sources of uncertainty are the following (list derived directly from [63]):

1. Accuracy of the surface surveillance systems at an airport
2. Final approach trajectory uncertainty
3. Taxi-in time uncertainty
4. Ramp entry/exit time uncertainty
5. Turnaround time uncertainty
6. Push-back time uncertainty
7. Uncertainty of departure queues and taxi-out
8. Runway crossing uncertainties
9. Uncertainty on the time an aircraft lands on the runway (wheels-on)
10. Taxi-route uncertainty
11. Departure and arrival runway prediction uncertainty
12. Uncertainty of departure runway balancing
13. Wheels-off time uncertainty
14. Gate-in time uncertainty
15. Runway occupancy time
16. Impact of ATCFM regulations
17. Impact of deicing in departure times
18. Impact of departure times on en-route sector times and en-route planning

The weather conditions present at the airport can also be thought of an uncertainty source and has a direct influence on the sources presented above. Using historical surveillance data from the Detroit Metropolitan Wayne County Airport, the authors performed 4 quantitative analyses. The first analysis focused on studying the ramp exit time uncertainty. Their analysis showed 23% of the departure flights is stopped at the ramp spot[3] while the remaining did not. It was also found that the ramp spot waiting time is about seven times longer if a flight stopped at the ramp spot. The second analysis focused on the taxi-route uncertainty. They concluded that the taxi-out time is correlated to both the distance between gate and runway as well as the route taken. Their analysis revealed that the shortest taxi route is not always the most optimal and often longer routes are used in order to avoid areas of congestion or crossing busy runways. The third analysis looked at the speed at which aircraft taxi the taxiway system. In addition to the aircraft type and single or double engine taxi, the taxi speed is dependent on the route intent, meaning whether a turn is imminent or the pilot continues on a straight segment. Based on the data, distributions were generated and filtered according to whether aircraft had to turn or continue straight at the following intersection. They showed that flights which continued straight after the intersection reached a higher average speed when entering the intersection than flights which had to perform a turn. The same conclusion was drawn by Gong [39] in a different study. Lee and Balakrishnan [52] also investigated the taxi speed uncertainty using fast time Monte-Carlo simulations. Their objective was to study the impact of aircaft taxiing at different speeds on the ground delay. They showed that the total ground delay increases as the taxi speed increases for both arrival and departing flights. They argue that this increase is expected because the taxi speeds of the flights are

[3] The location of the airport surface at which the aircraft enter (or exits) the ramp area.

constrained by slower flights. Their last analysis investigated the magnitude of delays experienced during a runway reconfiguration. Delays ranging from about 8 minutes up to 46 minutes were observed.

It is evident that uncertainty is an integral part of airport surface movement operations. A distributed airport control system designed to manage surface movement operation should demonstrate reliability under these circumstances. As will be discussed in Chapter 3, previous research in this domain studied the behavior of such a system under the occurrence of runway reconfigurations, a phenomenon often occurring at Schiphol. However, there is still plenty of room to explore the suitability of such methods under real-world conditions.

3

Previous Research in ATO

This study is part of ongoing research performed at the Air Transport Operations department of TU Delft, on the context of airport surface movement operations. The studies so far have looked at the problem from an agent-based modeling perspective. Agent based modeling (ABM) is a technique which allows the modeling of interactions between agents in complex socio-technical systems and provides the user with the means to identify emergent behaviors and system wide effects which would not have been the case with other modeling techniques [76]. Furthermore, all studies have implemented decentralized control in which the decision making process, in this case the aircraft traffic control, is shifted to a local level. Decentralization provides the means to distribute the demand across multiple nodes of the system and avoids capacity bottlenecks [93]. Decentralization is implemented by placing local controllers at taxiway intersections where each of them uses local information to solve conflicts locally. In addition, different coordination and multi-agent planning (MAP) methodologies have been implemented such that the global goal of such a system is met. The focus in this chapter is to provide the reader with summaries of the studies that make up the research so far. Section 3.1 summarizes some of the work performed in a PhD study performed by Udluft [93]. Particular attention is given to the studies performed by Noortmans [60] in Section 3.2 and Fines [36] in Section 3.3 who performed their analyses on Schiphol airport, each for different research objectives. The combined outcome of these studies is a working ABM simulator of Schiphol airport which will be used as a baseline for this study.

3.1. Decentralization in Air Transportation - Udluft (2017)

The original ABM tool to simulate ground operations was developed by Udluft during his PhD work [93]. His work was motivated from the fact that centralized resources in the air traffic management system, such as air traffic controllers, are limited in terms of capacity. Air traffic controllers have a limited mental ability of processing information, interacting with aircraft and making decisions. In order to ensure safe and stable operations, their amount of workload is dependent on the amount and complexity of traffic. Approaches to address the capacity bottleneck of the air transport system were proposed at the time but few were based on decentralization. Decentralized networks had already been proven to be more robust, agile and resilient than centralized networks and implementations were found in other domains, such as biology, robotics and business. One of Udluft's goals was to demonstrate the feasibility of using decentralized air traffic control to perform airport surface movements operations.

For his study a dummy airport with a simple symmetric layout (shown in Figure 3.1) was considered. It contained 3 gates, 18 intersections and 2 runways with two entries each. Decentralized control was implemented by placing local agents on the intersections of the taxiway system, on gates and also on runway nodes. Two types of agents were defined: the aircraft and the intersection agent. The aircraft agent's task was to maintain separation from other aircraft and react to heading and stop commands given by the intersection agents. The intersection agent's task was to control the traffic on adjacent taxiway segments. It continuously computes the shortest path for the aircraft under its control using Dijkstra's algorithm and is also responsible for reserving adjacent links such the aircraft can be safely accommodated. The agent has knowledge regarding the traffic density in its environment which is limited by the scope of information. To model the kinematics of the aircraft as accurate as possible, average values based on observations from one week of track data at Schiphol were used. The study only considered departing traffic.

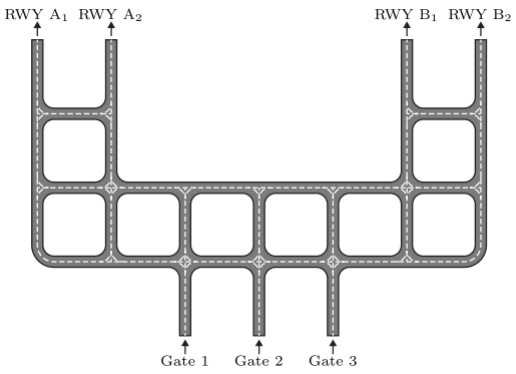

Figure 3.1: Taxiway layout in the baseline model. Retrieved from [93].

Upon running the model, a randomly generated flight schedule assigns an aircraft a gate and a runway. Once the aircraft leaves the gate the intersection agents are responsible for guiding it to its destination. The intersection agents coordinate between each other by means of an auction mechanism about whose aircaft gets to use the taxiway system first. ATC agents who have aircraft under their responsibility are allowed to bid for the allocation of taxiway segments. The highest bidder is allocated the taxiway segments which are subsequently given to the aircraft under his control. Udluft looked at the following aspects:

- **The impact of the scope of information:** A higher scope means that the intersection agents use more information regarding the state of the system. No performance gain was observed when using global information. In fact, local knowledge of the state of the system was sufficient to achieve global performance. From this it was concluded that local knowledge is sufficient for implementing decentralized control.
- **The impact of the scope of coordination between the intersection agents:** A higher scope means that the intersection agent is able to coordinate with other intersection agents further away. The results indicated that the performance of the system increases with increasing scope of coordination. The auction based coordination mechanism was compared to coordination by means of a procedure which aims at decoupling traffic streams on the airport layout. The procedure based approach achieved a higher performance with respect to measures like taxi time and total number of stops made but is not able to adapt to changing conditions such as runway reconfigurations as good as the decentralized approach which is inherently more adaptive. Furthermore, for very high aircraft spawn rates it performed worse than the auction based coordination mechanism with a certain degree of scope.
- **The complexity and emerging behavior of the system:** It was shown that different traffic complexities exist with different coordination strategies. Three coordination strategies were investigated: no coordination, auction-based coordination and procedure-based coordination. The uncoordinated case resulted in the highest complexity while the procedure-based approach resulted in the lowest complexity. An analysis based on runway re-configurations showed that the decentralized controllers were able to handle the disruption but no significant results were found with respect to the different coordination strategies.

Udluft concluded that given limited amount of information decentralization is a viable approach to perform airport surface movement operations and is also able to respond to developing traffic conditions. In addition, the results based on the auction-based coordination mechanism indicated that it can potentially outperform pre-defined static procedures.

3.2. Agent-Based Modelling of an Airports Ground Surface Movement Operation - Noortmans (2018)

During his literature search Noortmans [60] concluded that linear programming, search algorithms, historical data and agent based modeling where among the techniques most commonly used in the domain of ground

operations. However, most of the studies he investigated based their results on modeling steady state operations. No research was found to focus on improving the efficiency of ground operations in transient states, for example caused by runway re-configurations. Many of these studies made assumptions which oversimplified the problem. For example, some studies only considered departing aircraft while studies based on linear programming and search algorithms completely missed the existing interdependencies between the aircraft. Only a few studies used observations to obtain an understanding of the underlying operations and procedures during ground operations. Almost no study validated the generated results with actual ground operation data. Lastly, the concept of decentralized control was considered only by a few studies but has only been applied to simple airport layouts (referring to Udluft's work). Based on the above, Noortmans's focus was to obtain an understanding of the principles and mechanisms of decentralized air traffic control by comparing the emergent behavior of an agent based model to the actual ground operation.

Noortmans limited his scope into investigating the operations on the taxiway infrastructure. The apron operations were omitted and it was assumed that aircraft gates are always available and that no towing operations take place. The study was performed on AAS using a simplification of the actual layout, shown in Figure 3.4, a derived flight schedule, and an analysis of the strategies applied by ATCOs. Using 2 weeks of historical ADS-B data of ground traffic at Schiphol, he obtained the paths most likely travelled by aircraft as well as an actual flight schedule which he used as an input to his agent based model. The aircraft motion was modelled according to the fixed values used by Udluft. These are listed in Table 3.1.

Symbol	Description	Value
v_{max}	Maximum taxi speed	30 kts
v_{turn}	Maximum turn speed	10 kts
acc_{com}	Comfort acceleration	0.5 kts/s
dec_{com}	Comfort deceleration	-1.5 kts/s
dec_{max}	Maximum deceleration	-10 kts/s

Table 3.1: Specification of aircraft kinematics. Retrieved from [60].

Noortmans expanded Udluft's ABM simulator by defining a total of 8 agents in his ABM specification of which 7 are used for the realization of decentralized control (referred to as ATC agents hereafter). Detailed explanations of their properties and the interaction between one another can be found in [60]. The responsibilities of these agents are:

1. Source agent: Responsible for safely releasing aircraft agents into the taxiway system.
2. Sink agent: Responsible for removing aircraft agents from the taxiway system.
3. Apron agent: Responsible for accommodating the flow of aircraft agents entering and leaving the apron.
4. Runway agent: Responsible for managing the flow of aircraft willing to either take-off, land or cross the runway and is responsible to generate a schedule of future runway usage.
5. Endpoint agent: Responsible for slowing down the aircraft which are reaching their destination, either to line-up on the runway or enter the apron area.
6. Intersection agent: Responsible for controlling the aircraft by giving route, speed and stop commands.
7. Stopbar agent: Responsible for accommodating a safe runway operation by controlling the flow of aircraft willing to enter, leave or cross a runway as well as responsible for removing segments in the taxiway system.
8. Aircraft agent: Aircraft agents follow the commands issued by the ATC agents and are responsible for reaching their destinations as quickly as possible while maintaining separation from other aircraft.

The environment of the agent-based model consisted of a flight schedule which is observed by the source agents and a directed graph representing the taxiway system of Schiphol airport. Regarding the interactions between the agents, the following high level interactions were introduced on top of Udluft's implementation:

- Discussion and reservation making between ATC agents;

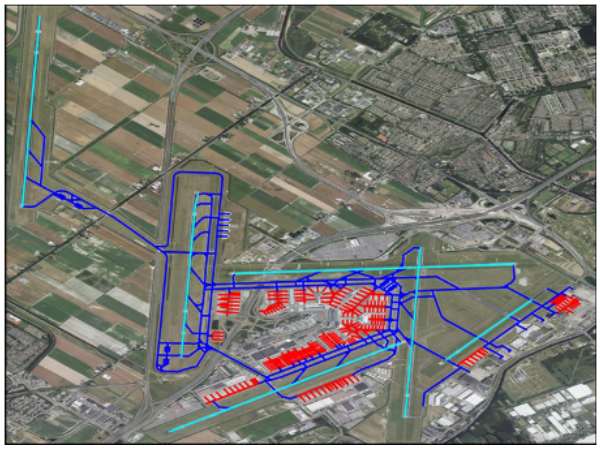

Figure 3.2: Original layout

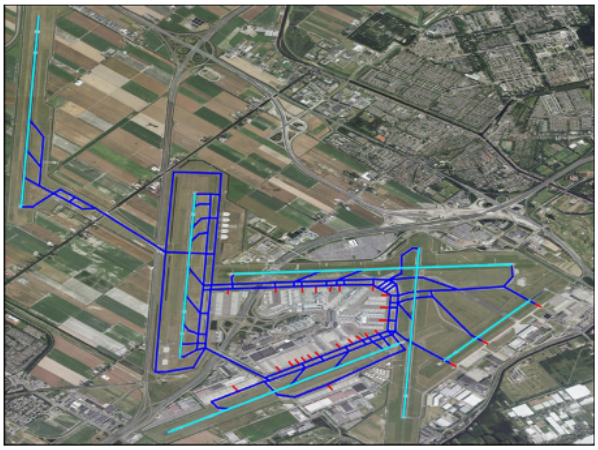

Figure 3.3: Simplified layout

Figure 3.4: The original layout on the left and the layout considered in the agent based model on the right. Retrieved from [60].

- Determination of future mode of operations;
- Sharing of runway schedule by the runway agents to the stopbar agents;
- Requesting permission to pass a stopbar;

The interactions between the agents and their environment include the reservation of taxiway segments. This is done in order to prevent other aircraft from entering taxiway segments used during an aircraft's hand off between ATC agents. Furthermore, ATC intersection agents are coordinated using an auction based coordination mechanism in order to achieve safe operations. This is similar to what Udluft used in his work. The auction is initiated whenever an aircraft is at a certain distance from its responsible intersection agent. The outcome of the auction process is used by the intersection agent to determine a future path for the aircraft under its control. After the extended ABM specifications were implemented on Udluft's model, the resulting model was fine-tuned based on observations and comparisons with actual operations.

The performance of the model was compared with the actual operations using a dataset containing 2 weeks of track data at Schiphol. The perfomance was measured using 4 key performance indicators, namely taxi time, taxi distance, average taxi speed and average traffic density. After conducting his experiments and sensitivity analysis on operational parameters as well as aircraft dynamics, the following conclusions were drawn:

- The performance of the decentralized control has similar patterns with the actual operations in terms of taxi time and taxi distance;

- There exist differences in the routing strategies between the simulated and the actual scenario due to the fact that decentralized control is using the taxiway system in a more flexible manner;
- The results supported Udluft's claim that a limited scope of information is required to allow safe operations but local information alone is not sufficient to achieve good performance;
- Decentralized control is able to handle more complex and chaotic operations since it does not need to comply to preferred taxi patterns and thus has a larger option space to respond to disturbances;
- The results supported Udluft's claim that the effect of coordination only becomes visible for highly congested taxiway networks;

3.3. Decentralized Control for Resilient Airport Surface Movement Operations - Fines (2019)

Fines [36] performed his research on the context of surface movement operations from a resilience standpoint. His objective was to evaluate the contribution of agent-based distributed planning and coordination to the resilience of airport surface movement operations when runway reconfigurations take place. The approach undertaken consisted of introducing a distributed planning mechanism to execute the tasks of the local ATC controllers. This is different to what Udluft and Noortmans used in their works where the agents' action were coordinated through auctions. The new approach enabled not only the coordination of activities but also the planning of those. He used a distributed version of the Conflict Based Search (CBS) Multi-Agent Path Finding (MAPF) algorithm and 2 variants based on highways to perform the planning activities. CBS is a type of algorithm where the individual agent plans, in this case paths, are coordinated only after they have been generated. Other planning algorithms do not follow this logic. CBS is explained in more detail in Subsection 5.6.3.

Planning and coordination

A distributed version of the CBS algorithm was implemented in his research. It works by detecting conflicts between agents bound to occur within a time window and resolves them by delaying or re-routing one of the agents. In addition to this, Fines also experimented with 2 adaptive highway mechanisms which were built on the original CBS implementation. This was justified from the fact that highways are used both in the resilience of city evacuations as well as in the current ground operations at Schiphol. Point-merge (PM) highways were used to merge aircraft paths which follow the same traffic flow and direct them towards specific points. The conflict-based (CB) highways mechanism on the other hand was used to create highways based on the amount of anticipated conflicts in regions of the airport's surface. In both cases, the highways were created and removed when certain traffic conditions existed in the environment.

ABM specification

The study was performed having in mind the concept of an Advanced Surface Movement Guidance and Control System (A-SMGCS) called Follow-the-Greens. Follow-the-Greens is a concept in which the airfield ground system controls the movement of aircraft in a decentralized manner. More information can be found in [36, 37]. The ABM was defined based on a simplified socio-technical representation of the system. In terms of model environment specifications, the same airport layout and derived flight schedule with Noortman's model was used. In terms of agent specifications, changes to Noortman's model had to be made. A total of 4 type of agents were defined. These are described in short together with their properties. For a more detailed description please refer to Fines's Msc thesis [36].

1. **Entry/Exit agent**: Responsible for creating initial routes for the aircraft, safely releasing those into the taxiway network and removing those from the taxiway system. They are located at all pier entry/exits and runway holding points. Agent properties:
    - Checks the flight schedule;
    - Generates an aircraft route based on Dijkstra's algorithm;
    - Removes aircraft from the taxiway network;

2. **Aircraft agent:** Responsible for following the commands of the ATC agents, taxi along their assigned route while maintaining separation from other aircraft. Each agent has a certain flight type depending on whether its departing or arriving and its destination within the airport environment. Properties:
    - Follows acceleration, heading and decision making protocols;
    - Follows speed control commands given by ATC agents;
    - Computes distance travelled;
    - Computes taxi time;
3. **ATC agent:** Responsible for detecting and resolving anticipated conflicts, creating and removing highways and guiding the Aircraft agents to their destinations. It is through the ATC agents that the CBS algorithms and its variants are implemented. Properties:
    - Detects anticipated conflicts;
    - Determines the type of the anticipated conflict;
    - Resolves a head-on type of conflict;
    - Resolves a crossing type of conflict;
    - Issues speed control commands;
    - Creates/removes point-merge highways;
    - Creates/removes conflict-based highways;
    - Removes edges opposite to the Aircraft agent's direction of travel;
    - Adds/ removes runway crossing related edges;
    - Hands-over the responsibility of an Aircraft agent to neighboring ATC agent.
4. **Airport Operation Status agent:** Responsible for determining which runways are in use and communicating that to the ATC agents which they should add or remove edges such that runway crossing can be avoided. Properties:
    - Determines the runway use based on the flight schedule;
    - Determines which edges close to the runways must be added or removed;

Conflict detection and resolution

The distributed version of the CBS algorithm is implemented in the following way. ATC agents detect conflicts by evaluating the time that it takes for Aircraft agents to reach their location. If Aircraft agents are anticipated to cross the ATC agent within a time window, T_{window} then a conflict is declared. The prediction of the time point at which an Aircraft agent crosses is made using a simple forward simulation of the aircraft's path. The algorithm computes the unimpeded taxi time of an aircraft by considering its dynamics when taxing in straight and turn segments. It serves as an approximation of the actual taxi time but does not take into consideration the dependency of an aircraft's route on other aircaft and other ATC agents. Once a conflict is detected it is further assessed to determine whether its a head-on or crossing conflict. If it is a head-on conflict the aircraft furthest away from the conflict node is re-routed. If it is a crossing conflict, the aircraft furthest away from the conflict node receives a speed control command computed using the estimated time of arrival to the node.

Results & conclusions

The contribution of the 3 distributed planning and coordination mechanism was assessed using two metrics, the average taxi time and the average taxi distance travelled. Resilience was measured by the change in the average taxi time and distance during and after the occurence of re-configurations. The results were compared with the corresponding values obtained from real-world operations. Fines used a total of 8 days of real flight data for his simulations from the same dataset as Noortmans used in his own research, resulting in a total of 6852 movements. The following results were obtained:

- **Taxi time & distance behavior:** The average taxi time was in favor of the 3 distributed mechanisms. CBS was the best performing mechanism with an average reduction of 1.07min/flight. In terms of the average taxi distance, no significant differences were found.

- **Resilience behavior:** During the period of runway re-configurations, no significant difference was found in the taxi times compared to the real-world. In terms of the average taxi distance, all 3 mechanisms demonstrated a lower performance than the real-world. The performance of mechanisms in terms of taxi times however outweighed the perfomance of the real-world after the occurrence of runway reconfigurations. No significant results were found in terms of the taxi distance. The most resilient behavior was achieved by the CBS+PM mechanism with an average delay of 1.5min/event compared to 2.12min/event in real-world.

3.4. Research gap

From the studies conducted so far there has been a significant contribution to the understanding of whether airport surface movement operations can be performed autonomously. The results indicate that, under certain assumptions, the distributed planning and coordination mechanisms are capable of managing ground movements to a degree which is similar to the real-world. In certain cases, they even surpass the perfomance of actual controllers. Taking as a basis the latest research on the subject, the goal of this Msc thesis is to improve the performance of the current implementation.

One aspect that has been thought of improving the perfomance of the current model is the use of a learning mechanism. So far there has not been any other research who studied the effects of distributed learning in controlling ground movements at an airport. Research in multi-agent learning however has received a lot attention during the last decade. The reason is that multi-agent systems (MAS) often operate in environments which have a complex and dynamic nature. For such systems it is unfeasible to specify the optimal agent behaviors at the design phase [77]. A learning mechanism able to capture relationships between certain environmental factors and consequences of particular agent actions, might result in enhancing the efficiency and effectiveness of a distributed airport control system. Both Noortmans and Fines recommended the use of learning elements for enhancing the performance of their models. Another aspect which can be looked at is the choice of the MAP mechanism. The field of multi-agent path planning (MAPP) is broad and a very active lately and although CBS was shown to perform to a degree similar to the real-world other, more advanced, algorithms might exist. Furthermore, in his study, Fines modeled the motion characteristics of the aircraft using fixed kinematics (Table 3.1), meaning all aircraft had the same taxi speed and accelerations. In the real-world however, aircraft do not taxi in the same manner. The performance of the distributed planning and coordination mechanism has therefore not been assessed under uncertain kinematic conditions. As mentioned in Section 2.5, variability in terms of aircraft accelerations and taxi speeds has been observed. To be able to make generalized conclusions regarding the efficiency of a distributed airport control system, uncertainty factors which form a big part of the actual operations should be taken into consideration. Of course accounting for all factor is complex, nearly impossible task. The focus will be more on how such a system behaves when aircraft kinematics are not identical.

The issues presented in this section will be the focus of this MSc study. The remaining of this literature review is focused on finding ways to address these. In Chapter 4, opportunities were learning can be applied are discussed and a review of various techniques is made. Next, in Chapter 5 a review of MAPP mechanisms is made.

4

Learning Mechanisms for Surface Movement Operations

This chapter presents a review of learning mechanisms which are suitable to apply in the domain of airport surface movement operations. In Section 4.1, opportunities in which learning can be applied to enhance the agents' tasks are identified and the most appropriate to address in this thesis is selected. In Section 4.2, a research which uses learning techniques in an agent-based modelling set-up similar to the current implementation is presented while in Section 4.3 approaches used in the literature of taxi-time prediction are reviewed.

4.1. Opportunities for Learning

This section describes opportunities in which learning mechanisms can be applied.

4.1.1. Modelling Incoming Traffic

The idea is inspired by Dresner and Stone [26] who performed their research on multi-agent road traffic management. More specifically, they looked at ways of improving the efficiency of agents placed at road intersections at controlling road traffic. These agents are able to communicate with vehicle agents and are responsible for directing vehicles through the intersection by accepting or rejecting requests made by the vehicles. Their hypothesis is that a machine learning model which predicts future traffic states can lead to more well-informed decisions for the intersection agents. The predictions can be based on inputs such as time of day, day of week and traffic history. In addition, they propose that the behavior of intersection agents can also be improved with the use of reinforcement learning. Actions which increase the system's overall goal are rewarded more than other actions. This requires, however, the definition of all possible state action pairs which is a complex task when considering a multi-agent scenario [92].

4.1.2. Predicting Future Traversal Times

As discussed in Section 3.3 the accuracy of the planning mechanism is limited by its prediction ability regarding the timepoint at which conflicts are going to occur. The forward simulation acts as an approximation of an aircraft's route and does not take into account dependencies with other aircraft taxiing. For example, it can be the case that based on an aircraft current route, a conflict is incorrectly declared. The aircraft is then re-routed taking more time and distance to reach its destination. Vice versa, a potential conflict might not be detected resulting in a collision between aircraft. The T_{window} parameter can be adjusted such that conflicts are avoided at all cases, but doing so results in suboptimal solutions as aircraft would be re-routed more frequently. An alternative method would be to use a learning mechanism which makes predictions regarding the time that it takes to traverse a certain link and adapts itself based on the conditions present on the airport surface. This might improve the accuracy of the distributed planning and coordination mechanism resulting in improving the global goals of the system.

4.1.3. Highway Generation

Fines [36] investigated the contribution of highways in performing resilient airport surface movement operations. Currently, highways are generated based on the amount of traffic and conflicts present on the taxiway network and removed based on a user defined parameter $t_{hwy-lifespan}$. A possible learning task would be to train a model able to capture the relationships, if any, between the location of congestion and anticipated conflicts, the time needed for highways to stay activated such that the disturbance is absorbed and the length of these highways.

4.1.4. Speed Profile Assignment

Another opportunity where learning can be applied is the allocation of speed profiles to the users of the taxiway network. In the current ABM specification, ATC agents issue a speed command to one of the conflicting aircaft such that a predicted conflicted is avoided. When no ATC commands are given, aircraft accelerate to their maximum taxi speed while maintaining the required separation with other aircraft. Speed commands could also be given such that congestion and build up of queues at intersections is avoided. A learning mechanism able to map congestion levels in the taxiway network to speed commands could potentially result in a smoother flow of traffic.

4.1.5. Conclusion

The perfomance of a distributed control system when coupled with one or a combination of the above learning mechanisms is certainly worth investigating. For the requirements of this MSc thesis however, it is decided to focus on a single learning task. The mechanism considered as a better fit to the current state of the model is the prediction of future traversal times. The reason is twofold. Firstly, the performance of the distributed planning mechanism, in this case the CBS, is dependent on accurate predictions of future conflict points. It is more logical to focus on methods which enhance the perfomance of already implemented mechanisms than introducing new ones. Secondly, the remaining learning mechanism are some what dependent on information regarding the location of aircraft on the taxiway network, and possibly their location at future time points. The errors present in the predictions of the simple forward simulation might be propagated and learned by the other learning mechanisms resulting in, potentially, an amplification of these errors. This will make the assessment of these other learning mechanisms not a straightforward task.

4.2. Anticipatory vehicle routing

A method in line with the objective of the selected learning mechanism was introduced by Claes [17]. Claes investigated the use of an anticipatory vehicle routing system to control road traffic. His adaptive MAS-based approach aimed at providing users of a road network with traffic information about which routes are more optimal to take. The idea is realised using the technique of delegated multi-agent systems (dMAS). At first the system collects information about the intentions of the road users. This information is used as input to an artificial neural network (ANN) model which form predictions as to how much time will it take for the user to traverse a certain segment of the road network. These predictions are then communicated back to the users.

Claes defined two types of agents in his MAS model. A vehicle agent and an infrastructure agent. The vehicle agent represents the road user and the infrastructure agent represents the traffic infrastructure. The interaction between the two is such that the vehicle agents provide information regarding which part of the infrastructure will they use (intend) and the estimated time of arrival and departure for each segment on that route. The information is shared only to the infrastructure agents involved in the vehicle agent's route. The infrastructure agents then provide the vehicle agents with predictions of the traffic conditions based on the amount of intentions that have been shared with them at any given time. This allows the latter to determine the most suitable route to follow.

The vehicle agent evaluates possible routes by dispatching the so called smart messages or mobile agents. These agents resemble in behavior the behavior of ants in the swarm-based ant-colony technique but they do not make their own decisions. A mobile agent is sent to the first infrastructure agent who represents the first link in the route being evaluated and queries the agent about information regarding the link traversal time. The predicted link traversal time or the estimated time for departure for that link is then used as the estimated time of arrival for the next link in the route. The process continues until the mobile agent reaches the road user's destination. The estimated time of departure for the last link in the route is then used as the vehicle's predicted time to reach its destination. After calculating the estimated time of arrival at the destination, the mobile agent returns to the vehicle agent and shares the information. Having mobile agents

to handle the communication instead of sending direct messages to the infrastructure agents significantly reduces the overall communication costs and bottlenecks. Besides the mobile agents, the vehicle agents also dispatch the so called intention mobile agents. An intention mobile agent operates in similar manner to the basic mobile agent but additionally informs an infrastructure agent that the vehicles it represents intends to traverse its link between the estimated time of arrival and departure for that link.

The infrastructure agent receives intention information which contain a vehicle ID and the estimated time of arrival. Time is discretized in time intervals and for every time interval, the number of intentions are aggregated and stored. In addition, it is possible for the intentions of the road users to be changed at a later time interval, for example when a better route is found by the vehicle agents. For this reason the infrastructure agents require regular updates from the vehicle agents regarding the intentions of their road users. Intentions are set to have a certain lifetime. During this lifetime, if the intention is not reinforced with an update, meaning that the validity of the intention can not be proved, the intention is considered irrelevant and is discarded when its lifetime ends. In this way, infrastructure agents maintain more accurate information which will yield in more accurate predictions of future states.

The infrastructure agents use the intention information to train, online, a prediction model that the vehicle agents can then access indirectly through the delegation of mobile agents. The predictions are made using an ANN. The difference of this method compared to the simple forward simulation is that it considers the dynamics existing between vehicles based on intention information. These intentions describe the future states of the segments connecting infrastructure agents. The ANN, known to be universal function approximator [45], is used to generate a function approximation which maps the intention levels into link traversal times. The inputs to the ANN are the current intention levels for future timepoint t as well as the intention levels of previous timepoints. Furthermore, in order to make the predictions more realistic, congestion information from downstream infrastructure agents are used as inputs. The reason is that link traversal times depend on the amount of inflow and outflow movements in the link. By monitoring the speed and vehicle inflow infrastructure agents can learn a threshold of intention level after which their link becomes congested. The information of whether their link is congested can be passed to the agent upstream via a direct message and be used as an input for their predictions. Like the intention information, the congestion information also has a certain lifetime. A visualization of the ANN structure is shown in Figure 4.1.

Claes evaluated the performance of the anticipatory vehicle rooting system under different traffic scenarios and compared it with that of a baseline scenario. Regarding the prediction ability of the system he concluded that the ANNs were capable of predicting the link traversal times in the network. The quality of these predictions however decreased as the prediction time horizon was increased.

The distributed control system presented in Section 3.3 shares a few similarities with the approach presented in this section. First of all, the vehicle agents and infrastructure agents have similar properties with the aircraft and ATC agents respectively in the current ABM specification. The ATC agents are located in parts of the airport's infrastructure just like the infrastructure agents are located in parts of the virtual road network. They have a view of their local environment and are able to communicate with agents upstream or downstream. Additionally, both need to make predictions regarding the future traversal times for the moving agents (vehicle or aircraft) although for different reasons each. The predictions made by the infrastructure agents in Claes's model are used to propose more optimal routes for road users. In this study, these predictions are used by the MAPP algorithm and its anticipated conflict detection and resolution property. Claes uses artificial neural networks to perform the mapping between intention levels and future traversal time. Generally speaking, other machine learning techniques could also be used for making the predictions. In the next section approaches found in the literature of taxi-time prediction are presented.

A possible implementation of Claes's approach to the current distributed control system is visualized in Figure 4.2. The steps are summarized below:

1. An ATC agent shares an aircraft's newly generated taxi plan to the ATC agents belonging to that path;

2. All ATC agents maintain intention levels based on received taxi plans. These are updated at frequent time intervals;

3. All ATC agents observe aircraft link traversal times and use these to train a model which maps current intention levels to observed traversal times;

4. When an ATC agents determines that an aircraft agent will pass by it (for example agent B), it sends smart messages which travel to every ATC agent involved in an aircraft's planned path (shown as agent

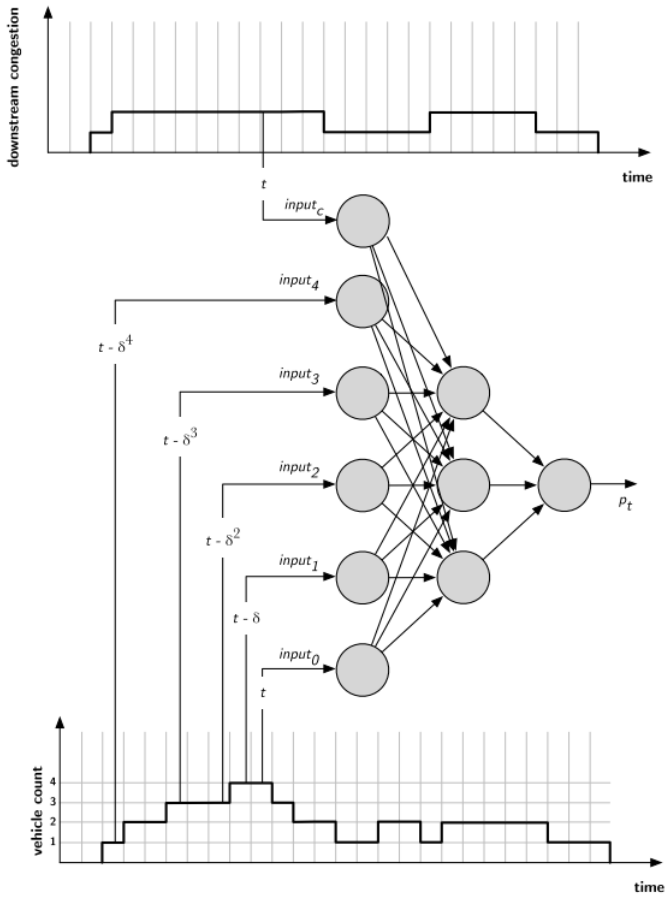

Figure 4.1: The link traversal time is calculated using the intention levels maintained by the infrastructure agent and congestion information from downstream agents. Retrieved from [17].

A), querying taxi time predictions for the links each represent. Aggregation of these estimates results in the predicted taxi time to the sender's location;

5. ATC agents compare aggregated taxi times and if a conflict is found, they act accordingly;

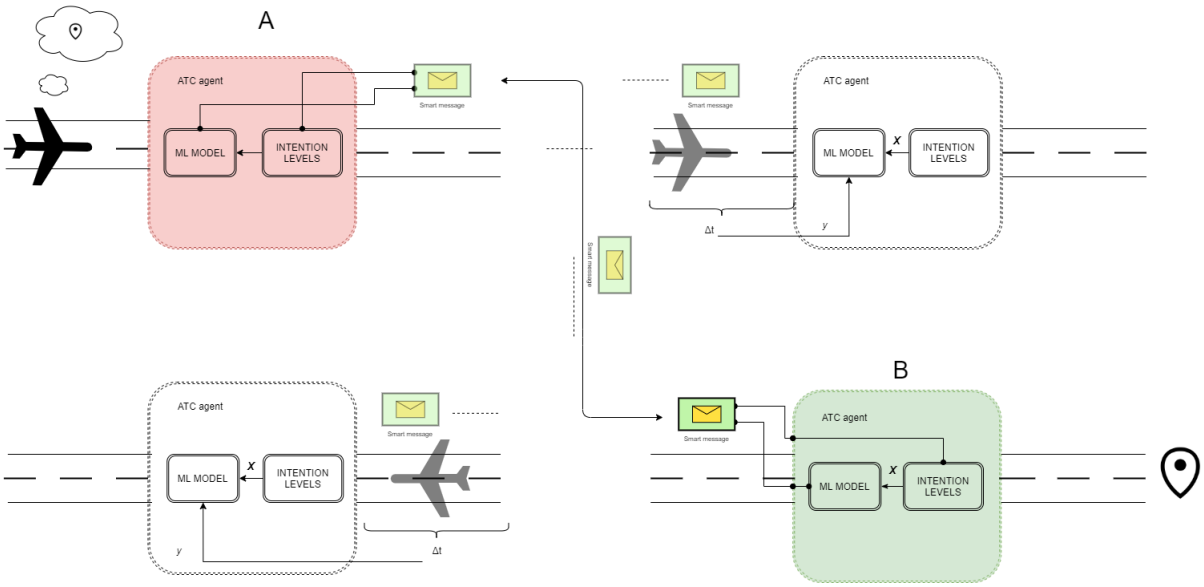

Figure 4.2: Concept of operations of the dMAS at an airport. Based on observed link traversal taxi times and intentions received by other ATC agents, agents maintain a ML model which maps intention levels to link traversal times.

4.3. Approaches for taxi-time prediction

This section presents the main approaches that have been used in the literature of taxi-time prediction.

4.3.1. Queuing models

Idris et al. [44] proposed a method based on a queuing model for improving the accuracy of taxi-out predictions. Taxi-out refers to the time duration between the actual push-back time to actual wheels-off from the runway. First, they analyzed 3 months of data from Boston Logan International Airport to identify the main factors that influence the taxi-out time. Their analysis showed that runway configurations, the airline terminal, the downstream restrictions and take-off queue sizes are the main factors. From these factors the take-off queue size was the most influential. In their study, queue size was defined as the number of take-offs that take place between the aircraft's pushback time and take-off. The queue size for a particular flight was determined by:

$$Q = N - N^P + N_P \tag{4.1}$$

Where N is the number of aircraft present on the airport surface at the pushback time, N^P is the number of aircraft that pushed back before the reference aircraft's pushback and took off after the reference aircraft, lastly N_P is the number of aircraft that pushed back after the reference aircraft's pushback and took off before the reference aircraft. Initially, only the relationship between N and taxi-out time was evaluated. Adding Q resulted in a more accurate trend line while the aircaft type did not seem to have a significant influence. The taxi time was then modeled using the Q for every runway or airline configuration possible. The queuing model was compared to a 14-day running average model on a test set of 1 month. The mean absolute errors obtained were 5.69 minutes for the running average model and 4.56 minutes for the queuing model.

A limitation of the proposed method and the running average model is that they are based solely on historical data. They are not able to adapt to the dynamic conditions present on the airport surface which the training dataset might have failed to capture. Furthermore, only the traffic flow of departing flights was considered in their study.

4.3.2. Reinforcement learning

Motivated from the fact that learning based approximate dynamic programming methods such as RL are adaptive and well suited for making predictions in dynamic environments, Balakrishna et al. [4, 38] applied RL methodology to test its effectiveness on making taxi-out predictions.

The method proposed by the authors uses RL to predict the taxi-out times of individual aircraft 15 minutes prior to gate push-back time. To construct the model the airport decision making process and the taxi-out

time prediction process were modelled as Markov Decision Processes. The RL model was trained, offline, using 6 months of historical flight data consisting of 10 features. A random set of 42 days was used to test the model performance. The prediction perfomance of the model was tested on flight data from a number of US airports including Boston Logan International Airport where the learning scheme consisting of 3600 states and 61 actions took 3 hours to complete. The authors do not provide any specifications of the machine on which the model was trained but they mention that the taxi-out estimate of flight scheduled to leave at t+15 min can be obtained instantaneously making their method suitable for real-time applications. The results obtained for different airports are shown in Table 4.1.

Airport name	**Within ± 3 min**	**Within ± 5 min**
Tampta International Airport	89.9%-95.7%	-
John F. Kennedy International Airport	-	20.7%-100%
Boston Logan International Airport	-	72.57%
Detroit Metropolitan Airport	89.9%-97.1%	-

Table 4.1: The prediction accuracy of the RL model when trained and tested on different US airports. Retrieved from [4, 38].

The adaptive nature of the proposed method is in line with the requirements of a distributed system aimed at performing surface movement operations. However, there is a major drawback with RL methods. According to [11], the main challenge when implementing RL in a multi-agent scenario is the exponential growth of the state-action space with increasing number of states and action variables. The complexity of RL methods is also exponential with the number of agents. In the Schiphol model (Section 3.3), several intersection agents are present which make the definition of the state-action space a complex task. In the study proposed here, actions only from 2 (ATCO and ground controller) agents were considered. A possible way of overcoming this limitation is to use a value function approximation method but this comes at the cost of prediction accuracy [91].

4.3.3. Multiple linear regression

Ravizza et al. [64] proposed a taxi-time estimation method based on multiple linear regression. Their study was performed using data from Stockholm-Arlanda Airport (ARN) and Zurich Airport (ZRH) and considered both departing and arriving flights. Both dataset contained information from an entire day's flight movements. For ARN the dataset contained a total of 661 movements while the ZRH dataset contained 679. In multiple linear regression the goal is to model the ith dependent variable y_i, as a linear weighted function of independent variables $x_{i1}, ..., x_{in}$ and an error term ϵ_i. The predicted value $\hat{y}$ for the ith observation is calculated using:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + ... + \hat{\beta}_n x_{in} \tag{4.2}$$

Where $\hat{\beta}_1, ..., \hat{\beta}_n$ are the estimated regression coefficients and can be estimated using least square regression. To measure how well the model fits the data, the adjusted coefficient of determination R^2_{adj} is used.

The authors did not predict taxi times directly but taxi speeds instead which, as they argue, are good estimators of taxi times. In an analysis performed, it was found that taxi speed estimations better fit the linear requirements of their linear model. The set of independent variables considered are the following:

- Distance that the aircraft travelled divided into three components;
- The total angular deviations between adjacent arcs on the route travelled by the aircraft;
- Whether the flight is departing or arriving;
- 8 integer variables (based on N and Q) inspired by the study of Idris et al. [44] which allow the consideration of the amount of traffic on the airport surface;
- For ZRH only, the operational mode of the runways;

Logarithmic transformations were used when necessary in order to fine tune the models. The R^2_{adj} values after fitting the models were 0.863 and 0.878 for ARN and ZRH respectively. The percentage of aircraft with a time difference between the predicted time and the actual time within a threshold of ±3min was 94.4%

for ARN and 95.6% for ZRH. The authors concluded that the average speed between the runway and the gate is highly correlated with taxi distance while arrival aircraft have a higher taxi speed than departing due departure queues forming at the runways. Lastly, the amount of traffic on the airport surface, the total turning angle as well as the runway operating modes also showed high correlations with the average taxi speed.

In a study performed by Atkins et al. [3] the same multiple linear approach was used to estimate taxi times at London Heathrow Airport, using a one week dataset. Rather than having a single regression model, the authors found it more useful to have separate models for departures and arrivals and to divide those according to which runway the aircraft were departing from or arriving at. They found that for departing aircraft the R^2_{adj} values were 0.903 and 0.956 for RWYs 27R and 27L respectively. For arriving aircraft the R^2_{adj} values were 0.812 and 0.861 for RWYs 27R and 27L respectively.

Kistler et al. [50] performed their analysis using data from 15 days of operations at Dallas/Fort Worth International airport (DFW). They developed different multiple linear regression models and used a variety of independent variables of which the most influential were the distance travelled, the number of stops and the surface traffic (both regional and total). Table 4.2 shows the setup of the different models considered in their study.

Model	Type	Surface Count	Stops Included	Variables	Adjusted R^2
LR	Linear	Regional	No	All	0.910
LR-S	Linear	Regional	Yes	All	0.968
LR-S*	Linear	Regional	Yes	Significant	0.968
LT-S	Linear	Total	Yes	All	0.967
LT-S*	Linear	Total	Yes	Significant	0.967
MR	Log-linear	Regional	No	All	0.997
MR*	Log-linear	Regional	No	Significant	0.997
MR-S	Log-linear	Regional	Yes	All	0.998
MR-S*	Log-linear	Regional	Yes	Significant	0.997

Table 4.2: Summary of the models including the type of variables used. Retrieved from [50].

Five of these models were tested on an independent dataset containing movements for one day of operations. The percentage of taxi-times within 1 minute from the actual taxi times are given in Table 4.3. The authors concluded that the linear models performed better than the log-linear models.

Model	Percentage
LR	58.11%
LR-S*	71.86%
LT-S*	71.07%
MR*	59.14%
MR-S*	64.55%

Table 4.3: Percentage of predicted taxi times within 1 minute of the actual taxi times for DFW data. Retrieved from [50].

4.3.4. Fuzzy rule based systems

Chen et al. [13] utilized the features identified by Ravizza et al. [64] to have a strong influence on taxi time predictions and trained a non-linear Fuzzy Rule-Based System (FRBS) using data from ZRH. They argue that FRBSs are more capable of capturing the non-linear, time-varying relationships present in the airport data than linear methods. Similar to ANNs, FRBSs have also been proven to approximate any real continuous function [100, 101]. FRBSs combine human expertise, sensory measurement and mathematical models to perform their predictions. The structure of a typical FRBS is shown in Figure 4.3.

A FRBS consists of five functional blocks. The rule base contains a number of fuzzy if-then rules, the database contains the membership function definitions used in the fuzzy rules, the decision-making unit performs the inference operation on the fuzzy rules, the fuzzification interface is responsible for transforming the inputs into linguistic values and the defuzzification interface is responsible of transforming the fuzzy results into an output [46]. The rule-base of a FRBS has the following form:

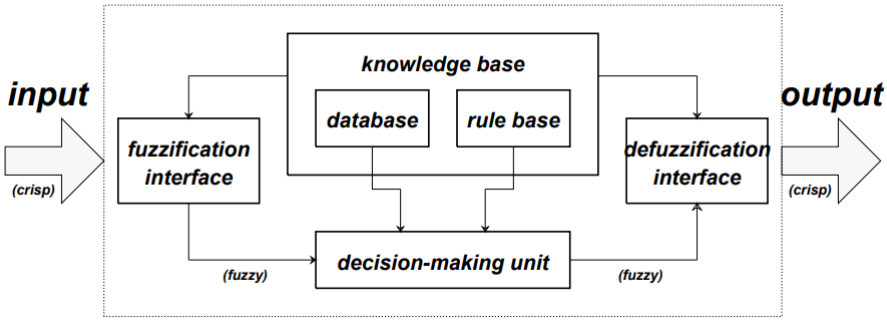

Figure 4.3: Structure of a typical FRBS. Retrieved from [46].

$$R_i: \text{ If } \quad x_1 \text{ is } A_i^1 \text{ and } x_2 \text{ is } A_i^2, ..., \text{and } x_j \text{ is } A_i^j \quad \text{then} \quad y_i = Z_i, \tag{4.3}$$

Where R_i is the ith rule to be considered, x_l (for $l = 1, ...j$) are the independent variables and A_i^l are the linguistic sets or fuzzy sets which are represented by a membership function $\mu_{A_i^l}(x_l)$. According to the authors, FRBSs have the following properties:

- Able to approximate complex nonlinear systems;
- Rules are able to differ in different regions;
- Human expertise can be integrated;
- Interpretability of the underlying system;

The authors used a revised version of the Mamdani FRBS in which a genetic algorithm and a k-means clustering algorithm were used to categorize the data into separate clusters and automatically generate the rules in the rule base. The membership function was chosen to have a Gaussian form.

The resulting Mambani FRBS model had 12 rules and was compared with a linear regression model. The results are listed in Table 4.4 showing a greater predictive perfomance for Mambani FRBS.

Model	**± 3min**	**± 5min**
Linear regression	95.6%	99.4%
Mamdani FRBS	98.8%	100%

Table 4.4: Comparison between the Mamdani FRBS and Linear regression on predictions made for ZRH. Retrieved [13].

In a subsequent study, Ravizze et al.[65] used another form of FRBS called Takagi and Sugeno's (TSK) FRBS. According to the authors, the TSK FRBS has the following characteristics:

- Each rule in the rule base resembles a multiple linear regression for a decomposed independent variable region;
- The rules work cooperatively in order to produce more accurate predictions;
- Some linguistic meanings could be lost in comparison to the Mamdani FRBS;

In TSK FRBS the if-then fuzzy rules are described as follows:

$$R_i: \text{ If } \quad x_1 \text{ is } A_i^1 \text{ and } x_2 \text{ is } A_i^2, ..., \text{and } x_j \text{ is } A_i^j \quad \text{then} \quad y_i = g_i(x_1, x_2, ..., x_j), \tag{4.4}$$

Where g_i is a linear function. The same membership function as in Mamdani FRBS was used, and a similar procedure for the generation of the rules in the rule base was followed.

Tests were performed using the ARN and ZRH datasets. For the former, 11 independent variables were considered while 15 for the latter. The TSK FRBS was compared with models based on: multiple linear regression, least median squared linear regression, support vector regression, M5 model trees and the Mamdani FRBS. A short explanation for the newly mentioned approaches follows below:

- **Least Median Squared Regression:** A linear regression (LR) approach which is more robust to outliers than multiple linear regression. Instead of minimizing the square of error, the median of these squares is minimized.
- **Support Vector Regression (SVR):** A supervised learning method which ignores training data above a certain threshold of the model prediction. It can be extended to a non linear model by transforming the training data into a higher dimensional space. The goal is to minimize the norm of the weights of the independent variables including the error for the training data which is further away from the prediction than the set threshold [81].
- **M5 Model Trees:** These are decision trees which store linear regression models in their leaves. The trees are constructed through a split process which occurs when certain criteria regarding the independent variables are met. The splitting process finishes when the standard deviation of the subset of training data is below a certain threshold or when it is too small [62].

Accuracy	Airport	LinReg	LMS	SMOreg	M5P	Mamdani	TSK
Accuracy within ± 1 min	ARN	54.28%	56.02%	57.86%	54.61%	58.21%	**58.80%**
	ZRH	58.38%	59.66%	**64.05%**	62.49%	62.97%	63.33%
Accuracy within ± 2 min	ARN	85.30%	85.18%	84.91%	85.19%	86.73%	**86.81%**
	ZRH	86.12%	85.99%	88.98%	88.15%	88.55%	**89.07%**
Accuracy within ± 3 min	ARN	95.40%	94.80%	94.32%	95.43%	95.72%	**96.16%**
	ZRH	95.55%	94.26%	96.60%	96.46%	96.54%	**96.89%**
Accuracy within ± 5 min	ARN	99.16%	98.81%	99.16%	**99.18%**	98.97%	99.08%
	ZRH	99.21%	98.56%	99.45%	99.46%	99.53%	**99.62%**
Accuracy within ± 10 min	ARN	**99.92%**	**99.92%**	**99.92%**	**99.92%**	**99.92%**	**99.92%**
	ZRH	99.92%	99.87%	99.97%	99.97%	**99.98%**	99.97%

Figure 4.4: Comparison of the models' predictive perfomance on ARN and ZRH data. Retrieved from [65].

The prediction accuracy for different time ranges is shown in Figure 4.4 after performing 15 repetitions of 10-fold cross validation. Highlighted in bold are the best results per performance measure for each airport. The results indicate that the TSK FRBS outperformed in many cases the predictions of the other regression approaches. This can be attributed to its ability of capturing non-linear relationships something which multiple linear regression, least median squared linear regression and M5 model trees can only do up to a certain extend. The support vector regression although also capable of modeling non-linear relationships, returned the worst results for the ±2 and ±3 minute accuracy in the ARN case.

4.3.5. Other machine learning methods

In a study performed by Lee et al. [53] various machine learning methods were used to estimate taxi times for departing aircraft at Charlotte Douglas International Airport (CLT). Studying surveillance data from departures performed in 2014, the authors identified the following features as the most influential for making the predictions: terminal location, gate, spot, runway, departure fix, aircraft model, weight class, taxi distance from gate to runway, scheduled pushback time, number of departures on the surface by runway, number of arrivals on the surface by terminal and unimpeded taxi times. The machine learning methods tested were the following:

1. **Linear Regression (LR)**
2. **Support Vector Machine (SVM)**: Or SVR for regression tasks. The radial basis function was used to transform the training data into a higher dimensional space.
3. k**-Nearest Neighbors (**k**NN):** A supervised learning algorithm used for both classification and regression tasks. It works by using a specified number (k) of training samples which are closest in the feature space to predict the value of the output. In their research k was assumed to be equal to 5.
4. **Random Forest (RF):** Belongs to the category of ensemble methods. The idea behind ensembling learning is that results from multiple weak algorithms can be combined via a voting scheme, in order to obtain better results on a specific task. For the case of a RF, the multiple algorithms are decision trees [10].
5. **Neural Networks (NN):** The NN architecture consisted on 18 nodes in the input layer, 3 hidden layers with the sigmoid activation function at each neuron, and a linear function in the output node. Although

not specified, it is assumed that every hidden layer has the same number of neurons. The training was set to 500 epochs.

According to the authors these are the most popular machine learning methods that have been used in estimating travel times for both aircraft and cars on road networks. For model training, the data was split into 4 categories based on traffic flow and weather condition: South/North-flow traffic and Good/Rain weather. The test dataset consisted of a day's operations. As a reference, a dead reckoning (DR) method where the predicted taxi time is equal to the unimpeded taxi time was used and compared with the models' predictions. The distributions of the predictions error for each model are shown in Figure 4.5 and the ± 5 minute percentage accuracy is listed in Table 4.5.

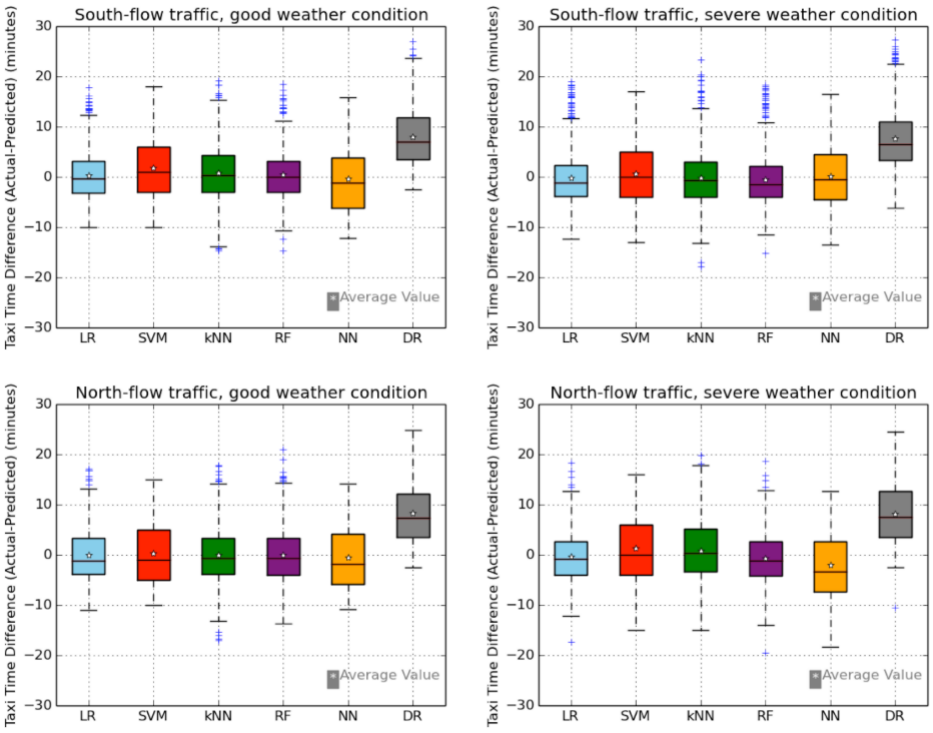

Figure 4.5: Predicted taxi time error distributions for different traffic flows and weather conditions at CLT. Retrieved from [53].

Test dataset	LR	SVM	*k*NN	RF	NN	DR
South flow, Good weather (15/08/2014)	**73.4%**	56.8%	65.8%	72.6%	48.9%	37.3%
South flow, Rainy day (12/08/2014)	67.6%	57.5%	64.4%	**69.4%**	53.1%	38.5%
North flow, Good weather (26/08/2014)	**67.0%**	56.4%	62.1%	64.9%	52.9%	34.8%
North flow, Rainy day (24/08/2014)	66.0%	55.6%	58.9%	**66.3%**	46.4%	35.3%

Table 4.5: Taxi time predictions within ±5 minutes for departing aircraft in CLT. Retrieved from [53].

The results indicate that all models had a greater predictive perfomance than the DR method. Looking at the distributions, the LR and RF show the smallest deviations from the actual taxi times. From Table 4.5 it can be seen that LR was particularly good at predicting taxi time when the weather condition were good while the RF predicted better on rainy days.

The fact that LR is one of the best performing methods can be attributed to a number of reasons. Non-linear, time varying relationships which are typical of airport operations according to [4, 38, 64, 65] might have not been present in the data. This can be a result of the choice of independent variables or the fact that aircraft

movements at CLT are not complex in nature. Another reason could be the absence of hyperparameter tuning in the SVM, *k*NN and NN which would explain their low perfomance. The authors have not indicated if they actually did perform it or not. All these model consist of parameters which specify their architectures and consequently their behavior during the training process. For example, with neural networks one needs to decide on the number of nodes on each layer, the number of layers and the activation function. Whether a regularization method is used to avoid overfitting can also be considered a hyperparameter. Simple LR on the other hand has no hyperparameters that require tuning.

Yin et al. [104] performed a study using both departure and arriving data from the Shanghai Pudong International Airport. Their method consists of formulating the machine learning inputs using a macroscopic network model. Such a model is able to describe the macroscopic resource flow at an airport. An example of such a model is shown in Figure 4.6 taken directly from their paper.

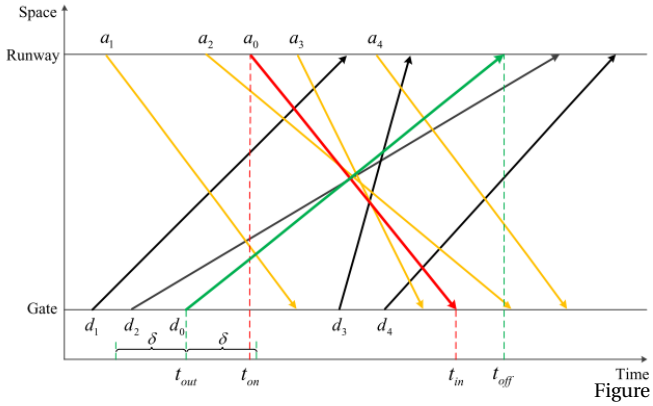

Figure 4.6: An example of a macroscopic network topology model of a taxi process. Retrieved from [104].

Predictors		Relevant aircraft	Values
SIFIs	D-SIFI	$\{d_1, d_2\}$	2
	A-SIFI	$\{a_1\}$	1
SCFIs	D-SCFI	$\{d_1, d_2, d_3, d_4\}$	4
	A-SCFI	$\{a_0, a_1, a_2, a_3, a_4\}$	5
AQLIs	D-AQLI	$\{d_1, d_3\}$	2
	A-AQLI	$\{a_0, a_2, a_3, a_4\}$	4
SRDIs	D-SRDI	$\{d_2\}$	1
	A-SRDI	$\{a_0, a_2\}$	2

Figure 4.7: The values of the machine learning predictors based on the macroscopic network topology model. Retrieved from [104].

Departures $d_1, ..., d_4$ and arrivals $a_1, ..., a_4$, have the following relationships with the reference departure aircraft d_0:

- d_1: "Off-block Before and Take-off Before"
- d_2: "Off-block Before and Take-off After"
- d_3: "Off-block After and Take-off Before"
- d_4: Off-block After and Take-off After
- a_1: "Landing Before and In-block Before"
- a_2: "Landing Before and In-block After"
- a_3: "Landing After and In-block Before"
- a_4: Landing After and In-block After

A set of ML predictors divided in 4 categories and 8 indices, depending on whether departing or arriving aircraft, were formulated (see Table 4.7). Surface instantaneous flow indices (SIFIs) denote the number of departures and arrivals when d_0 is being pushed back from its gate, surface cumulative flow indices (SCFIs) denote the number of departure and arrivals whose taxiing period coincides with the taxiing period of d_0, aircraft queue length indices (AQLIs) denote the number of take-offs and landing on the runway during the taxiing period of d_0 and finally slot resource demand indices (SRDIs) denote the number of pushbacks and landings in the departure slot $[t_0 - \delta, t_0 + \delta]$.

A LR, SVM and RF model were trained using one month taxi data and validated on a day's of operations. For error ranges of $\pm 1, \pm 2, \pm 3, \pm 4, \pm 5$, the RF model demonstrated the greatest generalization performance.

4.4. Conclusion

Literature supporting the realization of the task of predicting link traversal times was presented in this chapter. A study very close to this learning objective was performed by Claes [17]. His dMAS-based coordination mechanism was able to adapt to the dynamic conditions present at a traffic network, and make predictions regarding link traversal times. Route intentions which can be acquired from the users of the network contain information regarding future states of the system. The dependencies between the users are captured indirectly by a ML model which approximates a mapping function between current and historical intentions to observed values of link traversal times. This concept will be tested in this study as well.

For this purpose, a few modifications will have to be made in order to match the properties of aircraft agents as defined in [36]. More specifically in [17], the vehicle agents are responsible for sending the dMAS messages. This would be infeasible from the aircraft agents' side as new communication systems would have to be installed on-board. The alternative is to assign this responsibility to the ATC agents. In fact, this is more suitable and in-line with how the cooperative planning mechanism is currently implemented. ATC agents are already responsible for making taxi-time predictions (via forward simulations) and find conflict-free routes to aircraft agents. This means that only the properties of acquiring route intention information and using these to make taxi-time predictions need to be defined.

Although ANNs are used in [17] other ML methods, especially those able to capture non-linear patterns in the training data, might as well be suitable. A review on taxi-time estimation methods was performed. The review did not result in any clear winning candidate. The studies were performed using various airport layouts, modeling assumptions, types of features, amount of data as well as tuning methods. FRBS have demonstrated to have a good generalization performance but one needs to determine appropriate fuzzy rules first. Clustering and optimization methods can be combined to automatically generate those but this adds complexity to the entire the process. Linear regression methods in general are well suited for capturing linear relationships between variables. Other algorithms on the other hand like RFs, kNNs and SVRs seem more suitable to implement.

In order to measure the ability of the learning model to make link traversal predictions, it will be compared with the results obtained in [36]. The success of the learning algorithm, however, is correlated with the amount of data used and how well these represent real-like conditions (e.g runway reconfigurations). The 8 days of data (flight and runway schedules) used in [36] might not be sufficient to perform the training, validation and testing phases required for the successful training and evaluation of a ML model. Methods of increasing the amount of data will thus need to be investigated. One solution would be to gather a new set of track data from which a flight and runway schedule can be derived. These will be used for the training and validation phases of the learning model. The 8 days of [36] can then be used for testing. In case this approach is found to be infeasible, an alternative would be to create a synthetic dataset for training and validation, and similarly use the dataset of [36] for testing. To achieve a good generalization performance on test set, the synthetic data should contain flight schedules and runway configurations as realistic as possible. The statistical properties of such realistic scenarios could be investigated using the data in [36].

5

Multi-Agent Path Planning

As previously mentioned, airports can be characterized as complex, dynamic and unpredictable environments. Multiple users like aircraft and ground vehicles have to reach their individual goals while sharing a limited amount of resources such as runways, taxiways and gates. Some of these goals might even be conflicting with each other. To reach all individual goals in a safe and efficient manner it is important that the activities of all users are well planned and coordinated with each other. This is especially true for a distributed system aimed at providing conflict free and efficient paths along the airport's surface. The field of multi-agent systems (MAS) which deals with this problem is called multi-agent planning. Several algorithms have been introduced over the years. It is known that the success of these is highly dependent on the scenario that they are being used and that the applicability of every planning mechanism has its limits [27]. This chapter begins by providing information regarding the theory of cooperative MAP (Section 5.1). Due to the inherently distributed nature of tasks and systems in the problem of distributed airport taxiing operations, it is assumed that agents should cooperate between one another. Cooperative agents share their decisions with other agents and in the presence of a coordination mechanism, are willing to adapt their individual decisions up to a certain degree [84]. In Section 5.2 algorithms based on the A* algorithm are reviewed followed by rule based algorithms in Section 5.3. Next, in Section 5.4 hybrid mechanisms are discussed while in Section 5.5 reduction based approaches are presented. In Section 5.6 algorithms which are structured on two level are discussed. Last but not least, in Section 5.8 a trade-off between the presented approaches is made and the most suitable is chosen.

5.1. Background on Multi-Agent Planning

To properly define MAP, multi-agent coordination has to first be defined. Multi-agent coordination is defined as the managing of interdependencies between agent activities [55]. An interdependency is defined as the relationship between a local and a non-local task where carrying out one task affects the perfomance of the other [14]. In a typical coordination problem, the agents must decide on an appropriate set of actions to achieve their goals, in the presence of other agents' goals, distribute the limited resources available between them and execute their actions. Weerdt et al. [103] define the multi-agent planning problem as follows:

> *"Given a description of the initial state, a set of global goals, a set of (at least two) agents, and for each agent a set of its capabilities and its private goals, find a plan for each agent that achieves its private goals, such that these plans together are coordinated and the global goals are met as well."*

Based on a study performed by Durfee [28], Weerdt et al. identified the following phases towards solving a multiagent planning problem:

1. **Global task refinement:** Global goals and tasks are refined to the point where subtasks remain which can be assigned to individual agents;

2. **Task allocation:** Allocating of these subtasks to the agents;

3. **Coordination before planning:** Defining rules or constraints for the individual agents which prevents them from producing conflicting plans;
4. **Individual planning:** Making a plan for each agent individually so that it reaches its goals;
5. **Coordination after planning:** Coordinating the individual plans of the agents;
6. **Plan execution:** Execution of the plans and generation of results;

Additionally, the authors stress that not all phases need to be completed in order for multi-agent planning to be realized. For example phase 1 and 2 can be skipped if there is no common goal between the agents.

According to Durfee [27], three fundamental strategies in multi-agent planning problems are coordination before planning, coordination after planning and coordination during planning (i.e phase 4 and 5 run simultaneously). Coordination before planning aims at resolving all possible conflicts between agents before their local plans are constructed. This can be achieved in either of the three following ways:

1. Defining a set of rules that specify the allowed actions of the agents at specific scenario (social laws [79]).
2. Assigning tasks to agents that will not interfere with the goals of other agents (task assignment).
3. Adding constraints on the task assignment so that the resulting action will be conflict-free (coordination by design).

In coordination after planning the idea is to merge all local plans and schedules by taking into consideration all possible combinations and orderings and subsequently use coordination mechanisms to resolve the found conflicts, if any. The revised plans are then communicated to the agents. This approach yields in a somewhat centralized way of planning as a dedicated agent is responsible of gathering the plans of other agents and executing the coordination mechanism [78]. The CBS algorithm used by Fines belongs in this category. Lastly, coordination during planning entails the sharing of plans between the agents and the continuous re-planning of activities when conflicts are found. The continuous exchange of information between agents can be done in different levels of hierarchy (HTNs) [21]. Three directions are found in this coordination approach [78]:

1. **Centralized planning for decentralized plans:** Plans are created in a centralized way (i.e one agent creates sub-plans for each agents and monitors the progress). An example is the Partial Order Planning algorithm [6].
2. **Distributed planning for centralized plans:** Cooperative agents each contribute a part of their plan such that a global plan can be formed.
3. **Distributed planning for decentralized plans:** Agents have partial representation of other agents' plans and use these information to update their plans in view of improving the global plan. An example framework of this is the Generalized Partial Global Planning [24].

Multi-agent Path Planning (MAPP) is a sub-field of MAP which deals specifically with the planning of non-conflicting paths. Applications can be found in various fields such as robotics [95], computer games [80] and transportation [56]. The literature on MAPP contains a wide body of algorithms with varying perfomance and complexity. Some of these algorithms are explicitly designed to solve for path finding (PF) problems which are a subset of MAPP. A distinction between agent path planning and robot path planning is also often made. The following MAPP problems are found in the literature: Cooperative path finding (CPF) [29, 54, 80, 82, 83, 87], Multi-agent path finding (MAPF) [5, 7, 9, 18–20, 25, 34, 49, 72–75], Multi-agent path planning (MAPP) [16, 98, 99, 102] and Multi-robot path planning (MRPP) [68, 86, 96, 97, 105, 106]. Generally speaking, these approaches trade solution quality, completeness and scalability [98]. A commonly made distinction is between decoupled (or distributed) and coupled (or centralized) approaches [75]. In the former case the planning task is decomposed into independent problems for each agent and agents plan separately for their path. It is a fast and scalable approach but leads to non optimal solutions and in most cases incomplete. Decoupled approaches usually solve the MAPP problem in three phases [15]. First, individual plans are computed with respect to static obstacles and without considering the paths of other agents. In the second phase, agents are prioritized with respect to when their plans are to be restructured. In the last phase, the individual plans are restructured based on the priority list defined in phase two. In a centralized setting all agents are planned together. This can result in complete and optimal solutions at the expense of computational power because finding a solution becomes exponentially hard with increasing number of agents.

5.2. A* based approaches

Approaches which are based on the A* algorithm are discussed in this section.

5.2.1. Cooperative A* search

In his work on video game design, Silver [80] proposed three decoupled algorithms, all of which are based on the famous A* algorithm. Although the algorithms are tested in grid like environments, he argues that their applicability also extends in other path finding domains.

Firstly, he introduces the Cooperative A* (CA*) algorithm. In CA* single agent A* searches are first performed. After the path of each agent is determined, the states along the path are written in a 3-dimensional reservation table such that they are avoided in subsequent searches made by other agents. According to Silver, CA* is not able to solve certain classes of problems. The problem arises when a greedy solution of one agent prevents a solution for another agent. A variation of CA* is the hierarchical CA* (HCA*). This algorithm is similar to the CA* but uses the abstract distance heuristic to perform the A* searches. This algorithm executes a modified A* search in a reversed fashion. After a path is found the algorithm reserves the series of points of the path in the reservation table. Conflicts are thus avoided by disallowing agents to use paths already reserved by previous agents. The order by which agents reserve paths in the reservation table is chosen randomly [7]. The latter presents an issue in which the algorithm may not be able to find a solution for a given prioritization scheme. This makes HCA* incomplete.

Both CA* and HCA* compute paths before plan execution using a full depth cooperative search. In a scenario were the state space is large, their usability is limited. Silver introduced an online variant called windowed HCA* (WHCA*) to shorten the global search and to limit the prioritization issue of HCA*. In WHCA* cooperative path finding is performed within a limited user defined window (W), outside which all agents are ignored. Within that window a partial path is calculated for each agent and filled in the W-sized reservation table. Agents start following their partial paths and after a time interval, the window is shifted forward and a new cycle of partial paths is calculated. WHCA* returns less optimal solutions than the HCA* but is more applicable to real-time applications, since agents plan for the next W steps which significantly reducing the size of the reservation table. A downside of WHCA* is that it does not consider conflicts that might occur between agents. An agent might reserve W points without knowing whether these points are required by another agent or not. Like HCA*, WHCA* also suffers from incompleteness [7].

Byana and Felner [7] continued the work of Silver and introduced a variant of the WHCA* which takes conflicts into account. The algorithm is called conflict orientation WHCA* (CO-WHCA*). A important feature of the algorithm is that the reservations (or leftovers) of cycle $i-1$ are preserved in cycle i. CO-WHCA* has more flexibility in placing the window where the paths of the agents are reserved. This allows agents to be at most $W/2$ steps away from the conflict and agents who are not allowed to use the conflicting path have enough time to find an alternative. This is in contrast to WHCA* where agents have to plan right before the conflict occurs. The authors also presented a version of CO-WHCA* that uses a prioritization scheme (CO-WHCA*P) which prioritizes the agents that use the reservation table. For a given conflict the winner determination scheme, as they call it, considers all possible orders by which the reservation table can used. The agent that resulted in the lowest sum of costs is chosen as the conflict master and has priority over other agents. Tested on a map of Dragon Age Origin (DAO), a computer game used as a benchmark for testing path finding algorithms, both algorithms showed higher performance than WHCA* in terms of success rate and solution cost. CO-WHCA*P had the highest execution time since it needed to consider all possible combinations of agent prioritizations.

5.2.2. Standley's improvements

Standley [82] introduced two improvements to the standard A* algorithm for solving the CPF problem. Operator decomposition (OD) considers a representation of the state space in which each timestep is divided into the number of agents, so that each agent is considered one at a time. This distributed approach allows the A* search to reduce the amount of surplus nodes generated. These are nodes with $f > C*$ and which will never be expanded during the search. The method is able to achieve up to an exponential reduction in computing costs while determining conflict free paths but the technique is still exponential in the number of agents. Furthermore, due to the admissibility and completeness of the A* algorithm, Standley argues that A* with OD is also admissible and complete. The decomposition of every timestep, however, adds some complexity that may result in sub-optimal solutions. More specifically, there are situations in which agents are not allowed to move into spaces occupied by agents who have not yet been assigned a move in that state. The second

technique aims at improving the perfomance of the OD. Independence Detection (ID) works as follows. The algorithm first assigns each agent in a group and finds a path, using OD, for each agent independently. The found paths are then simulated. If a conflict is found during the simulation, a new path is determined for one of the conflicting agents which should not conflict with the original agent. If the process of finding a new path fails, it is repeated for the other conflicting agent. In case both searches fail, the agents are merged into a group and a path is planned for this group. All new paths are found using a conflict detection table which is updated regularly. The approach is also to achieve an exponential reduction in costs since it reduces the number of agents to be planned at any given time. Although both approaches reduce the computing costs, the resulting optimal OD-ID algorithm, as Standley notes, has still a computing time that is expensive for real-time applications [83].

The authors compared the performance of the OD-ID against the standard A* and Silver's HCA* [80] on a 32x32 grid map with random obstacles. Agents were spawned at random locations and given random destinations. The number of agents was chosen uniformly between 2 and 60 for a total of 10000 instances. They indicate that the algorithm is able to increase the performance of the standard A* algorithm by a considerable amount. However HCA* performs better than OD-ID on average in terms of both success rate and computational costs.

5.2.3. Approximate and optimal anytime algorithms

In a subsequent paper, Standley and Korf [83] proposed a complete algorithm, called MGS, that deals with OD-ID's drawback of having high runtimes. By dynamically removing the two constraints in the original OD-ID specification which make it optimal, the algorithm is able to trade optimality for computing time. The decision to drop the two constraints (or not) depends on a user defined parameter called maximum group size (MGS). The authors tested the perfomance of the MGS with different group sizes in experiments similar to the ones in [82]. For an experiment with 150 agents the results obtained are shown in Figure 5.1. MSG1 (MSG with group size equal to 1) solved 99.92% of the instances in under one second, while HCA* only 47.01%. The OD-ID was not able to solve any instance.

Later on, MGS was adapted to an optimal anytime algorithm (OA-MSG) which is more suitable for real-time applications. The anytime algorithm, as the name suggests, can be terminated at any time and the best computed solution up to that point can be retrieved. This is performed via a method called Iterative Deepening. The method enables the use of information computed in past iterations to be used in subsequent iterations which results in lower runtimes. The algorithm was tested against its optimal alternatives: the standard A* and the OD-ID. The performance curves are shown in Figure 5.2, The authors conclude that the optimal anytime algorithm can achieve a similar perfomance to the OD-ID and return a good quality of solution even when terminated earlier.

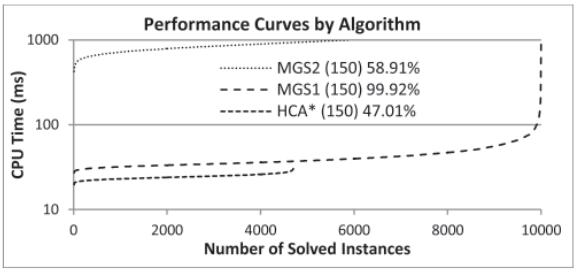

Figure 5.1: Performance curves for the approximate algorithms. Retrieved from [83].

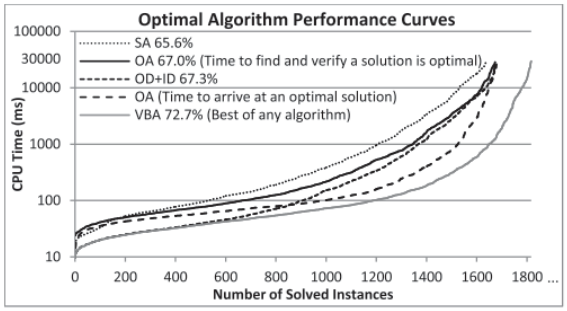

Figure 5.2: Performance curves for the optimal algorithms. Retrieved from [83].

5.2.4. Other approaches

In another study, Felner et al. [33] worked on improving the performance of the standard A*. They developed an algorithm that addresses the issue of surplus node generation present in A*. Standley's OD [82] shares a similar idea but his algorithm does not avoid all surplus nodes. Reducing the number of surplus nodes increases the computational performance of the search substantially. The algorithm proposed is called the Enhanced Partial Expansion (EPEA*) and is considered to be the best A*-based MAPF solver in 2014 [5]. Using

a mechanism called domain-dependent operator selection function (OSF), only the nodes with an f value smaller or equal to the optimal cost are generated. The authors tested their approach on two grid maps with sizes 3x3 and 8x8 and up to 8 and 12 agents respectively. Conflict avoidance was performed using a 2 level OSF. The results indicate that the runtime of EPEA* outperforms that of A* with OD by factor of up to a full order of magnitude. In addition, the EPEA* outperforms the enhanced version of ICTS, a tree based algorithm presented in Section 5.6.

Wang and Goh [102] introduced the guided iterative prioritized planning (GIPP) algorithm for solving a version of the MAPP which focuses on makespan. Makespan is defined as the number of time step required for all agents to reach their destinations [85]. The goal of GIPP is to move each agent at its destination in the shortest time possible while avoiding static constraints (obstacles) and dynamic constraints (agents). GIPP belongs to the category of local search algorithms. It analyzes every solution from a given solution space and applies local changes in order to produce an improved solution (or an optimal) within a certain time limit. The algorithm begins by running CA* searches to generate a series of optimal individual paths which then tries to optimize with respect to a cost function aimed at minimizing makespan. The method although shown to return makespan solutions similar in performance to WHCA*, it scales worse than WHCA* and Push and Swap (explained in Section 5.5). This is because it runs a centralized A* search which searches the joint agent space.

Choutan and Niyogi [16] presented a fully distributed complete multi-agent path planning (DiMPP) algorithm. In the distributed setting there is no centralized agent computing plans. Their method plans the agents' activities in three phases. In phase 1, each agent plans a path to its destination using the Fast Forward (FF) planning framework [41] by considering only static obstacles. In contrast to classical planning techniques such as the GPGP, FF incorporates heuristics in order to perform the search along possible states. More specifically, an algorithm called Enforced Hill-Climbing (EHC) is used to perform breadth-first search and find states with a better evaluation than the original state. The authors used the Euclidean distance between the current state and the goal state as means to evaluate the current state. EHC however is incomplete and might fail to reach a goal state. In case this occurs, a complete greedy best-first search is performed instead. Pruning techniques can also be used in order to reduce the search space of the algorithm. In phase 3, the possibly conflicting individual plans are restructured using a priority based decision making (phase 2). The decision making process is distributed among the agents and highest priority is given to agents with the longest plans. Every agent has limited information about their environment and other agents, an assumption commonly made in multi-agent systems. Through the exchange of messages, agent obtain full knowledge with respect to each others' plans. An example showing the priority decision making process is shown in Figure 5.3.

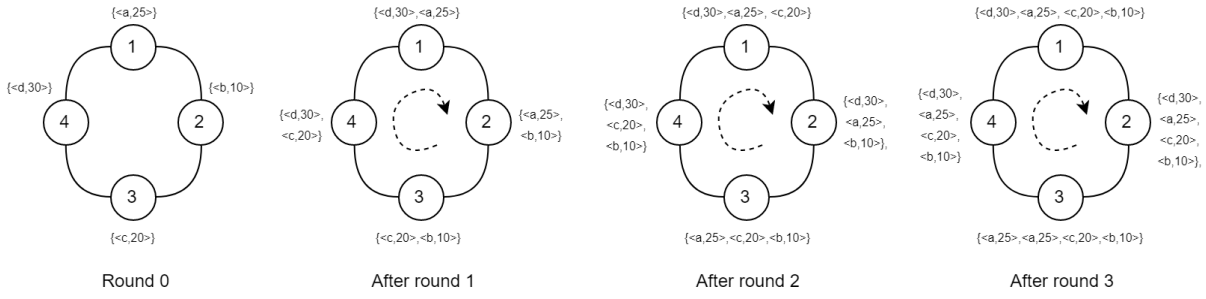

Figure 5.3: An example of the priority decision making. Agents (1,2,3,4) with identified paths (a,b,c,d) have lengths (25,10,20,30) respectively. At each round agents update their knowledge of other agents' paths.

The authors tested DiMPP on the brc202d DAO map which features similar properties to an airport's taxiway environment such as narrow corridors and bottlenecks. The algorithm was compared to the EPEA* and CBS. For small size problems DiMPP outperformed EPEA* in terms of both runtime and solution cost and it had a similar performance to the CBS. For large number of agents however, EPEA* and CBS scale exponentially whereas DiMPP scales polynomially.

5.3. Rule based approaches

Rule based approaches require certain rules in place in order to work. These rules are related to the properties of the agents, their environment and/or the interaction between them. Unlike the algorithms in the previous

section, rule based approaches usually do not involve an A* search to expand nodes. Typically, these methods return a solution relatively fast but often suboptimal in nature.

5.3.1. Push and Swap

Luna and Bekris [54] introduced a suboptimal algorithm for solving the CPF problem and named it Push and Swap (PS). The algorithm can be applied in problems with $n-2$ agents in a graph with n vertices. Two operations are used in the algorithm. During a $PUSH$ operation, an agent forces other agents to move away from its shortest path (by "pushing" them) and then proceeds with following that path. Certain scenarios are harder to solve and only pushing will not suffice. Such scenarios require agents to switch positions and this is accomplished with the $SWAP$ operation. $SWAP$ brings the two agents in a location of the graph which contains two empty vertices so that the swap can take place. While doing so other agents might have to move locations as a response to this. All agents are returned to their original positions after the swapping has taken place. Although the algorithm was initially shown to be complete on problems with 2 unoccupied vertices, De Wilde et al. [23] later on showed that its completeness cannot be guaranteed in certain scenarios. The algorithm was tested against Silvers's WHCA* [80] (with window sizes of 8 and 16) on a number of scenarios including a randomly populated grid with 20% obstacles. PS resulted in a higher success rate and lower computation times.

5.3.2. Tree-based agent swapping strategy

Khorshid et al.[49] introduced a tree-based approach for solving MAPF problems. They call it the tree-based agent swapping strategy (TASS). It is a rule based centralized algorithm which is shown to be complete only for tree graphs. TASS guarantees finding a solution in polynomial time but suboptimal in nature. The approach shares some similarities with the PS [54] algorithm presented in the previous subsection. Firstly, both of them perform a swapping operation, secondly they can solve only for certain graph topologies and thirdly they return sequential paths where one agents is moved at a time.

5.3.3. Push and Swap variants

A PS variant which returns solutions in which agents are moved in parallel was introduced by Sajid et al.[69]. Parallel Push and Swap (PPS) was shown to find solutions as fast as PS and of quality similar to the optimal anytime algorithm of Standley and Korf [83]. In a later research, DeWilde et al. [25] proposed Push and Rotate, a PS variant which deals with the drawbacks of the PS as presented in [23] and guarantees completeness in graphs with at least 2 unoccupied vertices. In the pre-processing phase of the algorithm, the graph is divided into subgraphs in which agents are allocated to them. Agents belonging in the same subgraph are allowed to perform swapping operations with each other. In the last step, agents are assigned a priority based on which they are planned. During the moving phase, a shortest path is computed for the agent first in the priority list and subsequently the agent is moved towards that path. At the event in which an agent is blocking the moving agent's path and the blocking agent has a lower priority, the latter is pushed to an empty vertex. Otherwise, a swapping operation is performed. The algorithm is also able to detect and solve instances in polygons something which PS failed to do. It does so using a rotate operation. Other rule based algorithms are presented in [8, 86].

An important thing to notice is that these algorithms assume that the agents are flexible to move in every direction. For example, in order to perform a swap operation agents must change their direction of movement within one step. This in turn makes their applicability to airport surface movement operations challenging. The infrastructure at a given airport might pose certain constraints in the movements that aircraft are allowed to do. A swap operation would require aircraft to be able to perform U-turns which is only possible in certain locations on the airport's surface. Reaching those locations would potentially increase the total travel distances and times, resulting in less efficient overall operations. The ruled based approaches are not as flexible as other approaches presented in this chapter and will therefore not be considered in the trade-off later on.

5.4. Hybrid approaches

Some approaches presented in the literature combine both searches and movement rules. These are characterized as hybrid approaches [75] and will be discussed in this section.

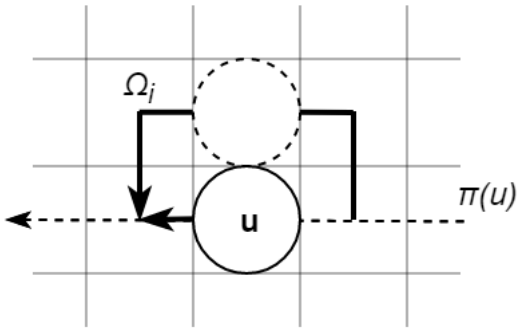

Figure 5.4: Example of an alternative path Ω_i to a precomputed path $\pi(u)$ of an agent u.

5.4.1. Flow Annotation Replanning

Wang and Botea [98] introduced the Flow Annotation Replanning (FAR) algorithm, a decoupled method for solving the MAPP problem by combining movement rules and independent A* searches. FAR abstracts a given grid map into a flow-annotated graph. In the new graph paths are flow restricted allowing movements only in single directions, just like in road networks. The directionality of paths is alternated between parallel paths. In addition, the algorithm imposes certain rules such as that the connectivity of the nodes is preserved. FAR is also an online algorithm meaning that planning and execution are interleaved. An optimal path for every agent is found by performing single agent A* searches. The searches do not take into account other agents on the map. Once an individual path is planned, the agent begins to follow it. The idea behind FAR is that by maintaining as many straight (one directional) paths as possible, the chance of having head-on and side-by-side collisions decreases. The algorithms favors straight paths by altering the standard A* search such that it gives priority on expanding nodes which yield in continuing straight lines. Agents are coordinated using temporal reservation tables similar to those found in Silver's [80] approaches.

The authors compare FAR's performance with an enhanced version of Silver's WHCA* algorithm [80] on a collection of maps from Baldur's Gate computer game. The experiments resulted with FAR having a better computational perfomance and scalability. FAR however suffers from two drawbacks. Firstly, FAR assumes a grid like structure which makes it not suitable for modeling an airport surface. Secondly, the approach does not consider what the consequences of a chosen path is on the paths available to future agents and may thus suffer from additional deadlocks [82]. Lastly, the approach suffers from incompleteness. It can not guarantee that a solution will be found within a certain time period [99].

5.4.2. Multi-Agent Path Planning

In a later paper, the same authors presented MAPP [99], a MAPF algorithm shown to be complete on certain types of instances which are named $slidable$. The algorithm begins by computing a path $\pi(u)$ for each agent to its target location while creating and caching alternate paths Ω along the way. A representation of such a situation is shown in Figure 5.4. Agents that are not slidable are placed last in a priority list. Based on their priority, each agent is then moved to its target location by following $\pi(u)$ and if necessary is pushed towards its alternative path. The latter occurs when a conflict with a higher priority agent is bound to occur. Once an agent reaches its destination, the algorithm moves the next agent in the priority list. The ordering of the agents in the priority list is chosen heuristically. A possible heuristic is the distance to the target location, giving a higher priority to agents that are closer to their goal nodes. By relaxing rule 1 and 3, the authors created two variations of the MAPP which are shown to achieve a higher perfomance than the basic MAPP.

In experiments performed against FAR and WHCA* for a collection of maps, MAPP was able to solve on average a higher amount of instances even for scenarios with 2000 agents. It did however require on average more computing time than its alternatives. Whether MAPP is suitable for the distributed aircraft taxiing task or not is difficult to say. Line 8 of Algorithm 1 in the paper requires agents to be able to move at any direction. Aircraft can perform U-turns only at designated locations or with the help of towing vehicles. It is therefore decided to not include this algorithm in the list of algorithms which will be compared later on.

5.5. Reduction based approaches

Approaches which aim to reduce the MAPF problem into a simpler problem which is then solved using other techniques are called reduction based approaches.

5.5.1. Constraint Satisfaction Problem

Real world maps typically are not built on random but usually contain underlying structures. Take an airport map for example. Long taxiway segments are normally placed parallel to the runways, and intersections are found in locations close to the terminals.

This what motivated Ryan [67, 68] into investigating a technique for reducing the size of search using domain information. The work he presented falls within the MRPP context. His method consists of exploiting the structure of a given problem and decomposing it in subgraphs such as stacks, cliques, halls and rings. A search using these subgraphs allows for a more informed pruning of the search space without sacrificing completeness. Subsequently, this new knowledge is encoded as a constraint satisfaction problem. Basically, the problem is encoded in integer variables over finite domains, and constraints which describe the relations between the variables that need to be satisfied. Variables are then assigned values and changes recorded are used to limit the domains of other unassigned variables. A prioritization towards the variable assignment can also be incorporated. This allows for more constrained variables to be dealt earlier in the search thus limiting the amount of backtracking when the assignments fail to satisfy the constraints. A variable assignment which satisfies all constraints presents a complete plan.

Ryan tested his approach on a map with 3 halls and 2 cliques. He reported the following. First of all, the problem decomposition into subgraphs combined with the informed type of search resulted in the highest rates of success. For easy problems the planner which considers map abstractions is 20-30 times slower than the planner without any abstractions (concrete planner). For harder problems, the abstract planner takes 0.25-0.30 of the time of the concrete planner. A prioritized variable assignment always yields a higher success rate but costs more to compute.

This approach can be considered suitable to be applied in airports as their lay-outs can easily be decomposed into subgraphs. However it has not been compared with other path planning algorithms. In addition, knowledge of declarative programming is required. CSP will therefore not be considered as an alternative.

5.5.2. SAT based solvers

Surynek [87, 88] introduced a different approach for solving the cooperative path finding problem. His approach is based on reducing the CPF problem into a boolean (or propositional) satisfiability problem (SAT). In such problems, the goal is to determine whether an interpretation that satisfies a given boolean formula exists. If it exists, it means that replacing the variables in the boolean formula by TRUE and FALSE will result evaluating the formula as TRUE. Such a problem is called satisfiable. Once the boolean formula is constructed, a SAT solver is used to find the solution. The challenging part is how to effectively encode a given CPF problem to a boolean formula $F(\Sigma, \eta)$. Surynek's work aims to answer this question by investigating several types of encodings [88]. To create a propositional representation of an agent's trajectory over time, Surynek uses Time Expansion Graphs (TEG). Put simply, a TEG is a graph representation in time where it captures all available movements of agents over a graph, at all time steps, up until a time point goal. The solution to the CPF is regarded as searching for non overlapping vertex disjoint paths in a TEG consisting of η layers. η is some makespan bound. An example of such a graph is shown in Figure 5.5.

An optimal makespan solution for CPF is obtained by querying $F(\Sigma, \eta)$ multiple times for different η and checking whether the formula is satisfiable. Queries are submitted to a SAT solver. As Surynek points out, a possible strategy for choosing makespan bounds is sequentially increasing η until an optimal makespan is reached. Reduction based algorithms such as SAT, typically suffer from high running times. There is a large overhead when reducing the CPF problem and the fact that the algorithm tries to return optimal solutions adds to that.

In a later research, Surynek et al. [89] presented a method for reducing the computing time of SAT based approaches. Their adapted SAT based approach, called MDD-SAT, focused on solving MAPF problems with respect to the sum of costs objective. MDD stands for Multi-value Decision Diagram (MDD), a type of data structure which was used to reduce the size of the TEGs. Instead of considering all states for all time steps as done in the TEGs, only the vertices and edges which result to valid paths are considered in the MDDs. According to the authors, this can lead to a reduction of variables in the model of up to two orders of magnitude. The approach was tested in grid maps and DAO maps against other optimal variants such as the EPEA*, ICTS and ICBS. The two latter are discussed in the following section. As can be seen in Figure 5.6, MDD-SAT is performing well both in terms of success rate and runtime when compared to its alternatives. In the brc202d DAO map which has similar properties to an airport's taxiway environment such as corridors and bottlenecks (shown in Figure 5.7), MDD-SAT still suffers from large overheads when compared to ICTS and ICBS. However, given enough time the algorithm will eventually reach the perfomance level of its alternatives.

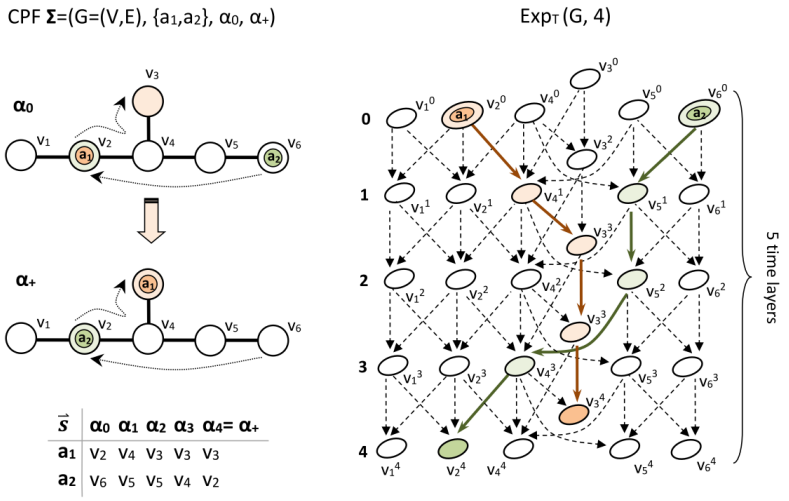

Figure 5.5: An example of a CPF problem and its expansion graph $Exp_T(G,4)$ consisting of 5 time layers. Retrieved from [88].

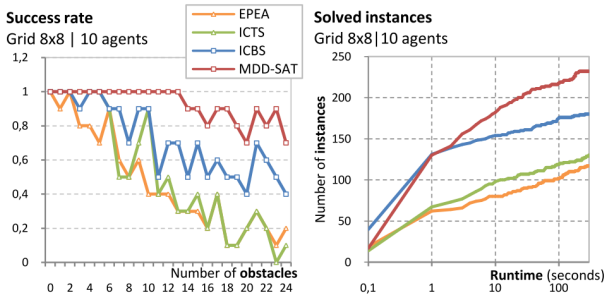

Figure 5.6: Success rate and number of solved instances on a 8x8 grid map with 10 agents. Retrieved from [89].

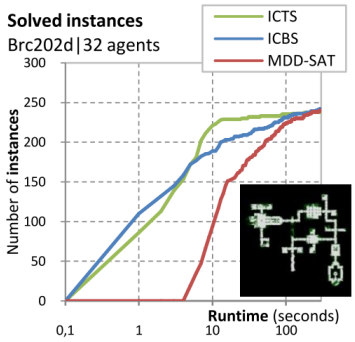

Figure 5.7: Number of solved as a function of runtime. Retrieved from [89].

Continuing in this research line, Surynek et al. [90] later proposed two suboptimal variants of the MDD-SAT, namely uMDD-SAT and eMDD-SAT. The former is the unbounded and the latter the bounded variant. The bounded variant returns a solution with cost less than or equal to $(1-\epsilon)\cdot C^*$, where C^* is the optimal solution and ϵ is a user defined parameter which specifies the degree of suboptimality. The unbounded variant simply returns any solution. To convert from optimal to suboptimal the authors relaxed one the constraints which makes up the original algorithm. eMDD-SAT was tested against the suboptimal Push and Swap [54] and ECBS [5] algorithms. ECBS is a bounded suboptimal variant of Conflict Based Search, a two-level MAPF solver presented in the next section. For tests on the brc202d DAO map, it was found that ECBS performed the best in terms of both solution quality and execution time. The performance graphs retrieved from their paper are shown in Figure 5.8. eMDD-SAT was compared with ECBS which is also bounded suboptimal for a suboptimality factor of 1.01. The results indicate a similar performance between the two, with the ECBS performing slightly better.

5.5.3. Integer Linear Programming

Yu and LaValle [105, 106] presented a method for optimally solving a multi-robot path planning problem using Integer Linear Programming (ILP). In their MRPP formulation, the authors assume that the agents can perform synchronized rotations in fully occupied cycles, something which is not assumed in the MAPF, or CPF definitions. The authors are motivated to use such a technique due the close relation of the MRPP with the network flow problems. Furthermore they argue that, in comparison to traditional A* based approaches, their ILP-based approach is capable of solving a greater range of problems and especially those with a high agent-vertex ratios. The authors also looked at reducing the size of the models given to the ILP solver using heuristics and therefore trade solution optimality to computing time. One of those heuristics divides the

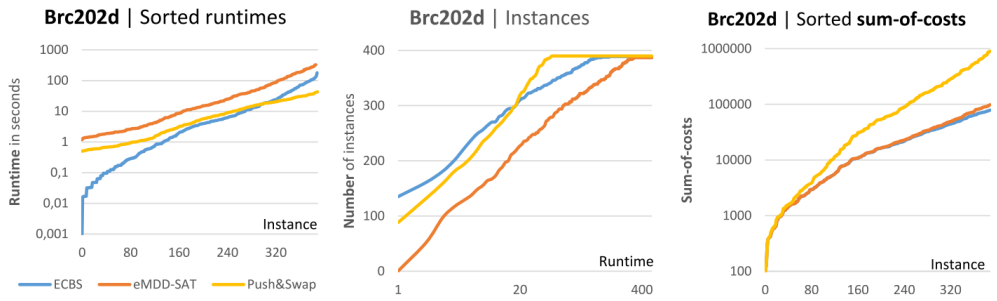

Figure 5.8: The performance of uMDD-SAT and other suboptimal alternatives. Retrieved from [90].

problem in k sub-problems in time. The makespan version of the ILP-based approach was compared against the Standley's optimal OD-ID [82] and Silver's suboptimal WHCA* [80]. On the worst test case, a 32x32 grid with 20% obstacles, OD-ID and WHCA* could not return solutions for more than 20 agents within a time bound of 600 seconds.

5.5.4. Answer Set Programming

Erdem et al. [29] used the technique of Answer Set Programming (ASP) to solve path-finding problems. Their approach is centralized, complete and optimal. ASP is a knowledge representation and reasoning paradigm which is based on logic programming. Using the so called answer sets (or programs), the solution space of a given problem is represented with logic terms and is constrained to desired outputs. An ASP representation of a problem is given to a solver just as done in ILP. The authors compared their approach to the ILP method in 180 randomly generated instances of 25x25 grid graphs with 0-40% obstacles containing either 10 or 20 agents. They report that the ASP approach performed worse in terms of success rate, computing time and cost of plans than the ILP approach presented earlier. However, the ILP approach demonstrated a larger performance decrease with increasing amount of obstacles and the memory requirements were lower for the ASP (<4GB) than the ILP (<10GB).

Although 4GB of memory usage is within the capabilities of most hardware these days, it confirms the poor scalability of centralized methods. Furthermore, the fact that certain instances took more than 15 minutes to be solved shows that these methods are also prone to large overheads. This makes their suitability of performing airport surface movement operation questionable. For this reason it is decided to not consider these methods as candidates for this study.

5.6. Two-level based approaches

In this section algorithms which are structured in two levels will be discussed. Usually at the top level a global search is performed and in the lower level the search is further refined. Algorithms from the M* and the CBS families belong in these category and a discussion of these follows.

5.6.1. M* family

In the context of MRPP, Wanger and Choset [97] proposed the M*, an algorithm which combines the properties of both coupled and decoupled approaches. On the top level decoupled planning is used to compute single agent paths using the A* algorithm. For the paths which are found to conflict at a later time point, a joint state space search (coupled planning) is performed again using the A* but for the conflicting agents only. So unlike A*, M* does not consider the regions of the spate space which have no conflicts. Furthermore, M* expands less nodes from the OPEN list than A* does. Similar to A* however, its computational cost increases exponentially with the number of colliding robots. The authors also show that M* is both optimal and complete. Following the idea of trading optimality for runtime, one can inflate the cost heuristic used in the M* by a value $\epsilon > 1$ and end up with the so called Inflated-M*. Recursive M* (rM*) is an optimal variant which improves M*'s perfomance when dealing with physically separated but simultaneously coupled sets of robots, resulting in a computational cost which is exponential not in the number of colliding robots but in the size of the largest set of mutually colliding robots. It does so by splitting the robot collision set maintained in the original M* into independent subsets for which planning is performed separately. The method is similar to the ID framework discussed in Section 5.2.2 but it does not keep the robots in the same set after a collision

is resolved as is done in the ID [96]. Running experiments on a grid with a density of 104 cells per robot and each cell having a 35% probability of being an obstacle, it was shown that inflated-recursive M* has the best peformance in terms of success rate, runtime and scalability.

In [96] Wanger presented the ODM* and EPEM* by replacing the A* with M* in the OD and EPEA* algorithms. When tested in a 32x32 grid with a maximum of 60 agents, the ODM* and EPEA* scored better in terms of success rate and runtime than A*, OD, M* and EPEA*. Their recursive variants ODrM* and EPErM* scored even better with the two of them having a very similar perfomance. Their suboptimal variants i-ODrM* and i-EPErM* showed an even more promising perfomance yielding higher success rates and runtimes. In terms of implementation i-ODrM* can be considered more straightforward to implement as opposed to i-EPErM* which is built on the EPEA* and requires the definition of a domain specific operator selection function (OSF).

5.6.2. Increasing cost tree search

Sharon et al. [72] developed a centralized two level framework called Increasing Cost Tree Search (ICTS) which solves MAPF optimally. An example of an ICT for three agents is shown in Figure 5.9.

The top level searches the ICT in a breadth-first manner. Every node s in the tree consists of a k vector of individual agent costs. The root (1st level) of the ICT consists of the optimal costs of the agents' paths which are computed assuming that no other agent exist along their ways. The second level of the tree consists of nodes (child nodes) in which a unit cost is added to the cost of one agent. An ICT node in which there is a complete non-conflicting solution in which the agent's a_i individual cost is C_i is considered a goal node. The total cost of a node is simply the sum of the individual costs. The low level checks whether a node s is a goal node. This is accomplished by storing all individual agent paths in a data structure called multi-value decision diagram (MDD). The cross product of the MDDs returns k non-conflicting paths for the agents. The ICTS was found to outperform Standley's OD-ID [82] framework based on A* in terms of both success rate and runtime when tested on the brc202d DAO map. The authors also presented a number of pruning techniques aimed at removing non goal nodes already from the high level so that the activation of the low level search is avoided. These techniques outperformed the basic ICTS in terms of runtime [73].

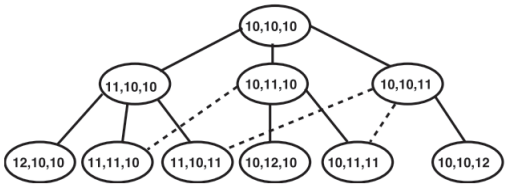

Figure 5.9: ICT for three agents. The dashed lines represent duplicate child node which can be pruned. Retrieved from [72].

5.6.3. CBS family

CBS

A state of the art algorithm for MAPF called conflict based search (CBS) was proposed by Sharon et al. [74, 75]. CBS can be considered as both a coupled and decoupled approach. It guarantees finding an optimal solution while the path-finding is done via single agent searches just like other decoupled approaches. Coordination in this context is performed through the merging of the individual plans.

CBS works on two levels. The high level searches the nodes of a constraint tree (CT) for conflicts via a best first search. A constraint tree consists of a set of constraints which contain information which prevent an agent from occupying a vertex at a specific time point, a solution which consists of all individual paths and a total cost which sums all individual agent path costs. If a conflict is determined in the high level, the node is declared as a non-goal node and is split into two child nodes, each having their own constraints. The nodes are then processed by the low level which tries to find paths for individual agent that are consistent with the newly assigned constraints. This is also done in a best first search manner. The new paths aim to avoid the conflict point either by making the agent move to an adjacent node or by making it wait to a current node. The authors used the A* algorithm to perform the single agent searches. After the node has been processed by the low level, a high level search is run again in order to validate the node. If after the validation no conflicts are found, the node is declared as a goal node and the solution is returned.

The authors tested the CBS on a number of DAO maps against other optimal algorithms. They concluded that the performance of the CBS depends on the structure of the environment. More specifically

in the brc202d map, shown in Figure 5.10, CBS was found to outperform both ODA* and ICTS with pruning (ICTS+3E).

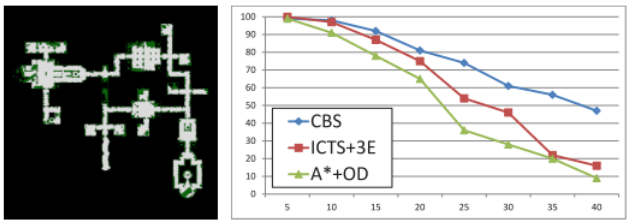

Figure 5.10: The success rate (y-axis) of CBS in the brc202d DAO map for increasing number of agents (x-axis). Retrieved from [74].

Meta-Agent CBS

A CBS-based framework was introduced by Sharon et al. [75] as a first step towards dynamically adapting algorithms. The authors note that the high level search of CBS is exponential with the number of conflicts encountered as opposed to the number of agents in A*-based approaches. This makes CBS to perform poorly in highly coupled situations. Meta agent CBS (MA-CBS) aims at improving this behavior by automatically identifying agents which are strongly coupled and merging them into a single agent instead of performing a split action. Once the merging is performed, the low level search is run for this meta-agent using any optimal MAPF solver. The decision to merge or split is defined as the merging policy. The authors use the number of conflicts parameter B to do this. If for example two agents have a B greater than a conflict bound, then these agents are merged into a meta-agent. If the conflict bound is set to 0, then algorithm behaves similar to Standley's ID framework (subsection 5.2.2). If on the other hand B is equal to infinity, the algorithm behaves like the basic CBS.

MA-CBS was tested under the same conditions as those which CBS was tested. The algorithm showed the most improvement in maps with open spaces. In airport like maps (brc202d) the MA-CBS with a conflict bound of 100, the framework had a slightly superior perfomance than the basic CBS. Figure 5.11 shows the success rate of MA-CBS and other optimal algorithms when tested on the brc202d DAO map.

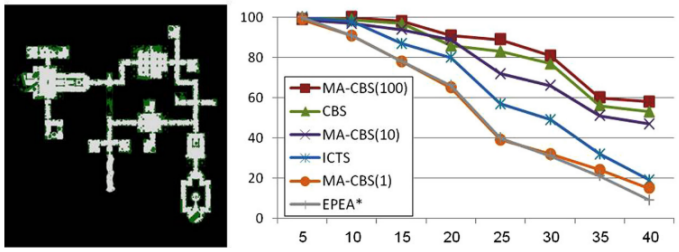

Figure 5.11: The success rate (y-axis) of MA-CBS(B) using EPEA* as the low-level solver on the brc202d DAO map for increasing number of agents (x-axis). Retrieved from [75].

Improved CBS

Boyarski et al. [9] presented a variant of (MA-)CBS called Improved CBS (ICBS) by introducing three new improvements to the basic implementation:

- **Merge and restart (MR):** When a merge decision is made for a set of agents inside a CT node, the CT node is discarded and the search is restarted from the root node. In the new search however the agents are merged from the beginning. This results in computational savings.
- **Prioritizing conflicts (PC):** The conflicts are classified in cardinal, semi-cardinal and non-cardinal and are hierarchically solved based on their class.
- **Bypassing conflicts (BP):** The split action is not immediately performed on the conflict node. It can be possible that the path of one of the agents is modified and therefore bypass the conflict. This reduces the size of the CT and saves a significant amount of search from being performed.

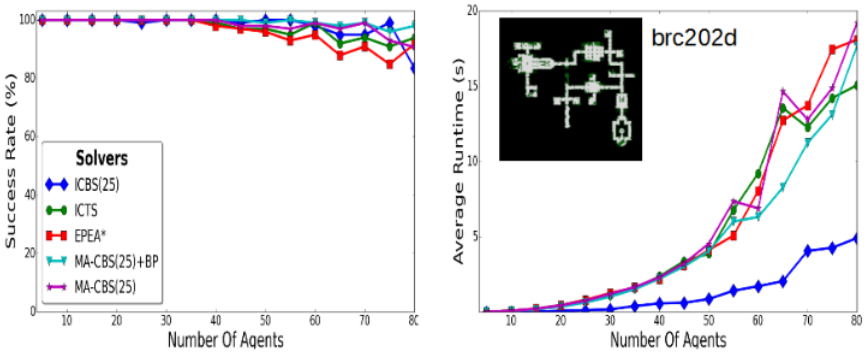

Figure 5.12: The success rate (left figure) and runtime perfomance (right figure) of several optimal algorithms when tested on the brc202d DAO map. Retrieved from [9].

The authors tested ICBS(25) (= MA-CBS(25)+BP+PC+MR) on DAO maps against other optimal algorithms like Sharon et al. [74, 75] did in their own work. The results for the brc202d map are shown in Figure 5.12. In terms of success rate, ICBS(25) performs somewhat similar to the other approaches. The benefit of the three improvements is well seen when examining the runtime performances. It takes less than 5 seconds in the worst case of having 80 agents for the algorithm to return a solution, outperforming by almost a factor of 3 the next best performing algorithm.

In a more recent study, Felner et al. [34] introduced ICBS-h, an enhanced version of ICBS. In the high level of CBS a best-first search on the CT is performed where the nodes are ordered by their cost. Nodes to be expanded and processed by the low-level are therefore prioritized based on their costs. The authors wanted to add admissible heuristics to the priority of the best-first search in order to make it more informed. Out of the four heuristics introduced, the ICBS-$h4$ was found to perform the best. When tested on the brc202d map against the basic ICBS, both resulted in a similar performance in terms of success rate. The former however had up to 2-3 times better performance in terms of runtime and number of nodes expanded.

Suboptimal variants

Barer et al. [5] introduced a number of suboptimal CBS variants. Optimality in CBS is guaranteed by running optimal best-first searches in both levels. The high level searches for the CT goal-node with the lowest cost, and the low level searches for an optimal single agent path that satisfies the agent's constraints. Nodes which have solutions very close to the optimal but not optimal are disregarded. This causes scalability and runtimes issues when the number of agents is high (see Figure 5.10).

Greedy-CBS (GCBS) is an unbounded suboptimal variant in which the high- and/or low level searches are relaxed, favoring the expansion of nodes which yield valid solutions fast. The degree of suboptimality is not specified, hence the term 'unbounded'. The high level search is relaxed by prioritizing CT nodes that seem closer to the goal node. To do so the authors developed a number of conflict heuristics h_c which allow the high level search to select a less conflicting nodes. Although the authors experimented with all the developed heuristics, results were only provided for the *number of pairs* heuristic which counts the number of pairs of agents that have at least one conflict between them. To relax the low level search a similar method was used. In the basic CBS the lowest level search return the shortest individual path that satisfies the agent's constraints. The authors adapted the low level search by using a best-first search instead, A* in this case, that prioritizes paths based on the value of the heuristic h_c. A path with a minimal value of h_c with previously assigned agents is then returned.

Three variants of GCBS were tested, namely GCBS-H which uses h_c on the high level and standard A* on the low level, GCBS-L where CT nodes are prioritized according to their cost in the high level and h_c is used in the low level and GCBS-LH which uses h_c for both levels. The success rate of the GCBS variants and the optimal CBS is shown in Figure 5.13. The GCBS-LH is shown to perform the best with solutions within 5% of optimal. No results in terms of its runtime performance are provided.

In addition two complete bounded suboptimal variants were introduced, namely Bounded-CBS (BCBS) and Enhanced-CBS (ECBS). Both algorithms use focal searches to return bounded suboptimal solutions. A focal search contains two lists of nodes: OPEN which the regular OPEN list of A* and FOCAL which contains a subset of nodes from OPEN. The FOCAL list uses two functions f_1 and f_2. f_1 is used to determine which

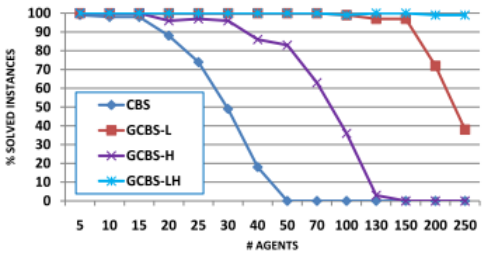

Figure 5.13: Success rate comparison between GCBS variants and optimal CBS. Retrieved from [5].

nodes are in FOCAL. Using a suboptimality factor w, all nodes n from OPEN which satisfy $f_1(n) \leq w \cdot f_{1,min}$ are contained in FOCAL. f_2 is used to determine which node from FOCAL to expand. If f_1 is admissible the returned solution is guaranteed to be at most $w \cdot C^*$, where C^* is the optimal solution.

In BCBS focal searches are used in both levels of CBS. In the high level a focal-search(g,h_c) is used to search the CT. $g(n)$ is the cost of node n and h_c is the conflict heuristic as described earlier for the GCBS. In the low level, a focal-search(f,h_c) is performed to find an individual agent path with satisfies the imposed constraints. $f(n)$ is the regular $f(n) = g(n) + h(n)$ which A* strives to minimize and $h_c(n)$ is the conflict heuristic. The authors prove that BCBS(w_H, w_L) is guaranteed to return a solution at most $w \cdot C^*$ for any values of w_H and w_L as long as $w_H \cdot w_L = w$.

The ECBS deals with the issue of how to best distribute w between w_H and w_L. ECBS is a w-suboptimal variant of CBS whose both levels are also focal searches. When a high level node n is generated, ECBS(w) performs a low-level focal search with OPEN list OPEN$^i(n)$ and FOCAL list FOCAL$^i(n)$ for the agent i affected by the constraint added in the high level [20]. FOCAL searches in ECBS are defined as follows:

$$FOCAL = \{n | n \in OPEN, cost(n) \leq LB \cdot w\} \tag{5.1}$$

Where LB is the lower bound of the optimal solution C*. Since all nodes in FOCAL are within w from the optimal solution, once a solution is found it is guaranteed to be at most $w \cdot C^*$. The advantage of ECBS over BCBS as authors explain is its additional flexibility in the high level once the low level finds low cost solutions. When compared to other bounded suboptimal CBS versions on the brc202d map (Figure 5.14, ECBS was able to maintain a success rate of 100% even with 70 agents while CBS could not solve any of the tested instances. This shows that relaxing the optimality of CBS and bounding it with a certain factor and additionally incorporating focal searches, significant improvements in success rates can be obtained. ECBS was also compared with previous introduced bounded suboptimal path finding algorithms, including M* variants and A*-based approaches. Figure 5.15 shows the success rates of the algorithms as a function of the number of agents when tested on a map of 32x32 grids, 20% obstacles and a suboptimality factor of 1.1. ECBS outperformed all algorithms. The authors did not provide any figures related to their respective runtimes but made the following conclusion in terms of the GCBS and ECBS. If the goal is to achieve solutions as fast as possible which can potentially be of high cost then GCBS is an ideal candidate. If on the other hand stability, reliability and solutions guaranteed to be bounded are of importance then ECBS is a more appropriate candidate. For surface movement operations where global goals include the reduction of the environmental footprint and taxiing delays, solutions not far from optimal are preferred.

In [18, 19] Cohen et al. introduced the ECBS with highways algorithm, ECBS(w_1) + HWY(w_2), which is particularly good when applied in Kiva-like domains. The algorithm works by finding paths for agents from user specified edges, called highways. Using an inflated heuristic with parameter w_2 in the low level search, the search is biased towards paths belonging to the user defined set of highways. As the authors state, the algorithm is $w_1 \cdot w_2$ suboptimal. In experiments performed with 150 agents in a Kiva-like domain, ECBS+HWY outperformed ECBS in terms of both runtime and solution cost. In a later paper Cohen et al. [20] developed iECBS(w_1). Similar to ECBS(w_1) + HWY(w_2) it uses a focal search with a highway heuristic but has only one parameter which makes tuning easier. When compared in a Kiva-like domain to ECBS with the same suboptimality factor, iECBS was found to have lower runtimes. In addition, two algorithms for automatically generating highways were introduced.

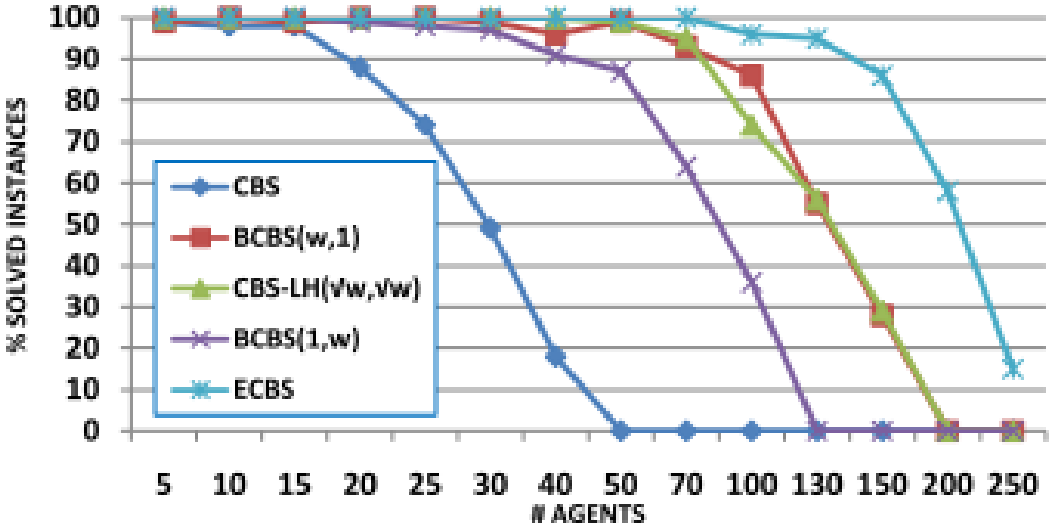

Figure 5.14: Tested on the brc202d DAO map. w = 1.01

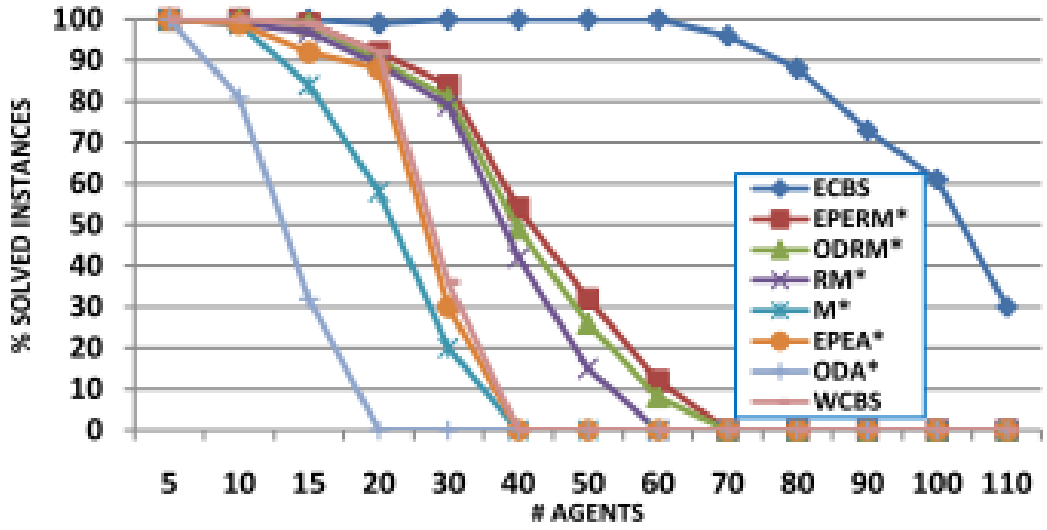

Figure 5.15: 32x32 grid with 20% obstacles. w = 1.1

Figure 5.16: Success rate of ECBS compared to bounded CBS versions (a) and other bounded suboptimal algorithms (b). Retrieved from[5].

5.7. Sampling based approaches

Sampling based algorithms are a class of motion planning algorithms that find conflict free paths by sampling points from a state space. Their ability of finding solutions quickly in high dimensional motion planning problems has made them quite popular in the motion planning community [12]. Among the most famous sampling based algorithms are algorithms based on rapidly exploring random trees (RRTs) [51].

The RRT algorithm iteratively builds a tree of states by randomly sampling states from the state space. When a new state is sampled, an attempt is made to connect the newly sampled state to the nearest vertex of the tree. If such connection is not possible, a new state is then randomly sampled. The algorithm runs until the goal vertex is reached. Once its reached, the edges of the tree are backtracked and the path from start state to goal state is returned. RRTs have been proven to be probabilistically complete, meaning that the probability of not finding a solutions tends to zero as the number of samples increases. A disadvantage of RRTs is that they do not make any guarantees regarding the optimality of the solutions. The problem was addressed by Karaman and Frazzoli [48] by introducing RRT*, an adapted anytime version of RRT which instead is proved converge to optimal solutions. The main difference with the RRT* is that when a solution is found, the algorithm does not stop but continues to draw new samples from the state space. The tree is therefore extended, new regions of the state space are explored and new low-cost paths are discovered.

Cáp et al. [12] introduced the multi-agent version of RRT* (MA-RRT*) which is able to solve CPF problems. MA-RRT* builds on top of Graph-RRT*, a modified version of RRT* suitable for motion graphs where agents move along. MA-RRT* samples the agents' joint state space in a uniform manner. The authors note that for sparse instances of CPF problems, the global solutions consist of paths which are usually similar to the optimal paths of the individual agents. To further increase the perfomance of MA-RRT* they proposed the informed-sampling MA-RTT* (isMA-RRT*) algorithm in which the sampling strategy is biased towards regions of the agents' state space which are close to the optimal single-agent paths. The isMA-RRT* algorithm works as follows: Single agent paths are first found using Graph-RRT*. Once the paths are found, MA-RRT* is run using a sampling function which draws samples from the Gaussian neighborhood of these single-agent paths.

The performance of MA-RRT* and isMA-RRT* was compared with A* and Standley's OA algorithm (introduced in Section 5.2). Tests were performed for varying grid sizes and numbers of agents. For these experiments a runtime limit of 5 seconds was set. The results indicated that MA-RRT* versions did not only solve instances quicker but were able to solve far more instances than the A* and OA algorithms. In particular, isMA-RRT* was able to solve 77% of the instances. The authors also looked at the suboptimality of the first valid solutions and best valid solutions, for the instances were either A* or OA found solutions. It was observed that when the algorithms were left to run, the best solutions converged to suboptimality factors very close to the OA.

Sampling based approaches form an alternative to the traditional search based approaches introduced earlier. isMA-RRT* in particular was demonstrated to perform well on grid-like maps when compared to the basic A* and Standley's OA. However, its performance is not yet quantified in maps which resemble an airport's environment like the brc202d DAO map nor has it been compared to other state of the art algorithms for multi-agent path-finding. For this reason it will not be considered when performing the trade-off between the approaches presented in this chapter.

5.8. Trade-off

The algorithms which are considered relevant for this research are listed in Table 5.1. An initial comparison is performed between them using the findings in the literature indicating their performance compared to the algorithms which they were tested against. Since not all algorithms are compared directly with one another, as a first step we select the best performing ones and compare these at a later stage of the trade-off process. The table is divided according to the tests performed in the literature. For comparisons between three algorithms a scoring system of [-1,0,1] was used, whereas for two algorithms [0,1] was used. In certain cases a comparison with respect to one of the criteria was not possible due to the lack of data in the respective papers. The algorithm with the highest score is selected and considered at a subsequent step of the trade-off process. The criteria selected to assess the suitability of the methods are performance and subjective based. These are:

- **Success rate:** Relates to the ratio of agents for which the algorithm was able to find valid conflict-free paths. An algorithm with a high success rate is able to successfully plan conflict-free paths for most of the agents in a considered simulation.

- **Runtime:** Relates to the computational time it took for the algorithm to produce a valid solution. Algorithms with low runtimes are more suitable for real-time applications such as distributed airport control and are thus scored the highest.
- **Solution cost:** Relates to how close the solution of a given algorithm is to the optimum. The lower the solution cost, the less steps are involved in an aircraft's paths and the lower the taxi distance and/or taxi time. Algorithms which produce low cost solutions are scored the highest.
- **Scalability:** Relates to how well the algorithms scale with increasing number of agents. It is usually a product of how well the algorithm's success rate and runtime is with increasing number of agents.
- **Implementation:** Relates to the complexity of a given algorithm. Complexity in this case is based on criteria such as the number of step needed to implement the algorithms and the ease at which they can be programmed. The lower the score the more difficult it is to implement it.

	Approach	Success rate	Runtime	Solution cost	Scalability	Implementation	Total	Test environment
[7]	WHCA*(16)	-1	-1	0	-1	1	-2	32*32 grid, 20% obstacles
	CO-WHCA*	***0***	***1***	***0***	***1***	***-1***	***1***	
	CO-WHCA*P	1	-1	0	-1	-1	-2	
[102]	WHCA*(16)	0	0	0	0	1	1	32*32 grid, 20% obstacles
	GIPP	***1***	***1***	***0***	***1***	***0***	***3***	
[16]	**DiMPP**	**-**	**1**	**0**	**1**	**0**	**2**	brc202d DAO map
	EPEA*	-	-1	1	-1	1	0	
	CBS	-	0	1	0	0	1	
[89]	MDD-SAT	-1	-1	-	-	-1	-3	brc202d DAO map
	ICTS	***0***	***0***	**-**	**-**	***1***	***1***	
	ICBS	***0***	***1***	**-**	**-**	***0***	***1***	
[90]	u-MDDSAT	0	0	0	0	0	0	brc202d DAO map
	ECBS	***1***	***1***	***1***	***1***	***1***	***5***	
[90]	e-MDDSAT	-	0	-	-	0	0	brc202d DAO map
	ECBS	**-**	***1***	**-**	**-**	***1***	***2***	
[74, 75]	EPEA*	-1	-1	-	-1	1	-2	brc202d DAO map
	ICTS + pruning	0	0	-	0	0	0	
	CBS	***1***	***1***	**-**	***1***	***0***	***3***	
[75]	ICTS	-1	-	-	-1	1	-1	brc202d DAO map
	CBS	0	-	-	0	0	0	
	MA-CBS(100)	***1***	-	-	***1***	***-1***	***2***	
[5]	CBS	0	0	0	0	1	1	brc202d DAO map
	GCBS-LH	***1***	***1***	***0***	***1***	***0***	***3***	
[5]	***OA-MSG***	***1***	-	-	***1***	***1***	***3***	32*32 grid, 20% obstacles
	GCBS-LH	0	-	-	0	0	0	
[5]	EPErM*(1.1)	0	-	-	0	-1	-1	32*32 grid, 20% obstacles
	ODrM*(1.1)	0	-	-	0	0	0	
	ECBS(1.1)	***2***	-	-	***2***	***-2***	***2***	
[5]	CBS	-1	0	1	0	1	1	brc202d DAO map
	BCBS(1.01)	0	1	0	1	0	2	
	EBCS(1.01)	***1***	***1***	***0***	***1***	***0***	***3***	
[9]	MA-CBS(25)	1	-1	-	-1	0	-1	brc202d DAO map
	ICTS	-1	0	-	0	1	0	
	ICBS	***0***	***1***	**-**	***1***	***-1***	***1***	
[20]	ECBS	-1	-1	-1	-1	0	-4	Kiva like domain
	ECBS+HWY	0	0	1	0	0	1	
	i-ECBS	***1***	***1***	***0***	***1***	***-1***	***2***	
[34]	CBS	-1	0	-1	0	1	-1	brc202d DAO map
	ICBS	0	-1	0	0	0	-1	
	ICBS-H4	***1***	***1***	***1***	***1***	***0***	***4***	

Table 5.1: Quantitative trade-off of multi-agent path planning algorithms.

For the best performing A* based approaches the following things can be said. Both CO-WHCA* and GIPP produce suboptimal solutions. The former is an distributed online algorithm while the latter is centralized. Although there has not been a direct comparison between the two in the literature, CO-WHCA* is considered to be more suitable since distributed methods scale better than centralized approaches. In addition, GIPP is more complex, since an optimization mechanism is applied after a solution to the path finding problem has been found. When comparing ICTS with ICBS, their scores based on [89] are equal. In [9] however ICBS scored higher than both ICTS and MA-CBS. However, ICBS scored lower than its most recent variant ICBS-H4 when tested on the same map. Taking these facts into account a qualitative trade-off on the remaining algorithms is performed in Table 5.2.

It can be seen that the two suboptimal variants of CBS, ECBS and iECBS have the best qualities among the compared algorithms. iECBS is a better choice if its applied to Kiva-like domains and makes more sense to choose if the introduction of highways is of interest. The concept of introducing highways however, adds an additional dimension to the learning task presented in Chapter 4. The reason is that the ML model would have to be trained on scenarios which include the existence of highways. In [36] highways are automatically

Approach	Advantages	Disadvantages
CO-WHCA*	- Scalability	- Tested in grid maps - Not complete - Worse performance than GCBS, ECBS @ 32*32 grid, 20% obstacles
DiMPP	- Scalability - Fully distributed - Complete	- Success rate worse than GCBS, ECBS @ brc202d, >20 agents
OA-MSG	- Optimal - Higher success rate than GCBS @ 32*32 grid, 20% obstacles	- Tested in grid maps
GCBS-LH	- Low runtimes - Always returns a solution	- Unbounded suboptimal
ECBS	- Good success rate up to 100 agents @ brc202d - Degree of suboptimality can be specified. - C++ code available on GitHub	- Runs faster than CBS but no exact figures are available.
iECBS	- Runs faster than ECBS in Kiva-like domains - Same suboptimality factor as ECBS	
ICBS-H4	- Low runtimes, 120-380 ms for 100 agents @ brc202d - Better memory requirements than CBS	- No implementation online - @ 60 agents has lower success rated on brc202d than ECBS

Table 5.2: Qualitative trade-off of multi-agent path planning algorithms. Red color is assigned to the unsuitable algorithms. Orange color is assigned to less suitable algorithms. Green color is assigned to the most suitable algorithms.

generated when certain conditions are in place. This can happen anywhere in the taxiway network. It is more logical to first assess the perfomance of the learning mechanism in the basic case where no highways are used. If the predictions are found to have a positive influence to the performance of the MAPP algorithm, then a subsequent step would be to expand the learning mechanism by adding the dimension of highways. Possibly in combination with one of the highway generation mechanisms found in [36] or one of the learn-based highways mechanisms introduced in [20]. For this reason the ECBS algorithm will be used in this study.

6

Research Proposal

In this chapter the research proposal for this MSc thesis is presented. In Section 6.1, the research objective and research question are outlined followed by a description of the tasks that need to be performed and relevant sub-question to be answered. In Section 6.2 the research scope is defined.

6.1. Research Objective & Questions

Previous research in the department of Air Transport Operation at TU-Delft has demonstrated that distributed control is a viable approach for managing the complex, dynamic and unpredictable nature of airport surface movement operations. Distributed control has been achieved by distributing the responsibility of controlling an aircraft to local controllers placed at multiple locations of the taxiway network. A more recent research investigated the contribution of a distributed system, based on multi-agent planning mechanisms, to handling aircraft traffic during and after runway reconfigurations. This MSc thesis aims to carry on with this research.

Two ways of improving the current implementation have been identified in this literature study. Firstly, the multi-agent path planning algorithm that the current system is based on can be extended. An enhanced version found in the literature, has been shown to outperform its predecessor in terms of scalability, solution quality and runtime. Secondly, the accuracy with which the conflict detection and resolution activities are being performed in the path planning algorithm can be improved. Currently, the prediction of the time point at which a collision might occur is performed by a simple forward simulation. This approach considers the amount of straight and turn segments along an aircraft's path as predictors. However, calculating the taxi time to any location deterministically serves only as an approximation of real ground operations, which are far more stochastic in nature. Factors such as variable aircraft kinematics and dependencies with other taxiing aircraft and agents, can affect the accuracy of these predictions. A prediction technique combining delegated multi-agent systems and supervised learning will be used to address these shortcomings. The research objective of this MSc is defined as follows:

> *To analyze the performance of a distributed airport control system consisting of a learning based cooperative multi-agent planning mechanism in managing airport surface movement operations.*

The research question linked to the research objective is the following:

> *To what extent can a distributed control system, consisting of a supervised learning-based cooperative multi-agent planning mechanism, improve the performance of airport surface movement operations with respect to indicators such as taxiing time and distance?*

1. **How can existing ABM specifications be extended to incorporate learning abilities**

   The first step towards answering the research question consists of implementing the learning based mechanism to the existing solution. This means that the conflict detection and resolution properties of the distributed CBS algorithm presented in [36] have to be adapted. The prediction of conflict

time-points will no longer rely on forward simulations of aircraft kinematics. Machine learning models trained online and maintained by the ATC agents will be used for this purpose. ATC agents will be able to perform predictions as well as query predictions from other agents. The means by which these queries will be handled is through a dMAS. To accommodate this aspect into the current model, the dMAS properties need to be specified and the ATC agents properties need to be adapted. Relevant sub-questions are the following:

(a) What changes in the original ABM specifications need to be made in order to incorporate the dMAS?

(b) What are the ABM specifications of the dMAS?

2. **How will the learning task be effectively executed?**

An important thing to note is that the learning phase of the machine learning model will be executed online, meaning that the model will be trained during simulation time. In addition, a decision needs to be made as to whether the prediction will be made by a single, centralized ML model maintained by all ATC agents simultaneously, or via a distributed version where each ATC agent trains its own model. After adapting the agent properties and defining the dMAS specifications, the next step involves the formulation of the learning strategy. This consists of three steps, namely data preparation, feature selection and model selection. In terms of data preparation, flight and runway schedules, which currently are the inputs of the ABM simulator, have to be prepared and formed in the required format. Regarding training data, two alternatives need to be investigated. Model training will either be performed using new data from which flight schedules and runway schedules will be derived, or using synthetic data resembling actual scenarios as close as possible. It is decided to use the 8 days of data in [36] to perform the tests required to measure the performance differences between the new model, the previous model and the real operations. The second step involves the feature extraction. According to [17], the main inputs to the ML model are intention level information which capture the future states of the system. This information will be transmitted via the dMAS. Besides current and past intention levels, more features such as those used in the taxi prediction literature (Chapter 4) can be used. If a centralized ML model is used, then the locations at which target values are observed also need to be distinguished. The last step involves the selection of a machine learning model. NNs, RFs and SVMs are among the algorithms commonly used in the literature to make travel time predictions in both airport and road related scenarios. The means by which the hyperparameters of the selected algorithm will be tuned, also needs to be investigated. Relevant sub-questions are:

(a) Will the data be acquired or generated?

(b) Which data will be used for training, validation and testing?

(c) Which predictors are of importance when predicting link traversal times?

(d) Which machine learning algorithm is suitable for this learning task?

(e) Which hyperparameter tuning strategy is going to be used?

3. **What is the contribution of the supervised learning mechanism to the performance of the distributed CBS mechanism?**

To answer this question, the performance of the supervised learning based mechanism will be compared to the current implementation. The comparison will be performed using the same data and performance metrics as used in [36]. More specifically, the average taxi time and taxi distance of the flights will be evaluated before, during and after the occurrence of runway reconfigurations. Direct comparison with the results in [36] will give an indication as to whether the predictions made by the machine learning model outweigh or not, the predictions made by the forward kinematic simulations. Additionally, the new method has to be evaluated in terms of its runtime and ability to make predictions in real time. Relevant sub-questions are:

(a) How do the predictions of the machine learning algorithm compare to the predictions made by the simple forward simulation?

(b) What effect does the machine learning algorithm have on the runtime of the system?

(c) How does the learning mechanism affect the performance of the CBS under steady state conditions?

(d) How does the learning mechanism affect the performance of the CBS when runway reconfigurations take place?

4. **To what extent can the learning-based CBS mechanism match the performance of the centralized air traffic control?**

   The ability of the learning-based CBS mechanism to perform airport surface movement operations will be compared with the real-world. Similar to the previous question, performance will be measured with respect to how well operations are handled before, during and after the occurrence of runway reconfigurations. Relevant sub-questions are:

   (a) What is the performance difference under steady state conditions?

   (b) What is the performance difference when runway reconfigurations take place?

5. **To what extent can the ECBS improve the performance of the learning-based CBS mechanism?**

   Once the learning-based system is implemented and evaluated, the next step will be to extend the CBS algorithm to its enhanced variant, the ECBS. To do so, the ATC agents' specification will have to be adapted. A key difference between the two algorithms, is that ECBS uses focal searches in both of its levels. Focal searches process nodes from the OPEN list of the A* and return solutions which are w-suboptimal. This is different from how optimal CBS is defined which returns the shortest individual agent path consistent will all imposed constraints [5]. This aspect needs to be addressed in the current ABM specification, since the distributed version of CBS uses Dijkstra's algorithm which has no heuristic evaluation. Once the ECBS algorithm is implemented, the resulting distributed control system will be trained on a number of simulation runs using a flight and runway schedule. The performance of the learned model will be compared with the real-world. Once again, performance will be measured with respect to how well operations are handled before, during and after the occurrence of runway reconfigurations. Relevant sub-questions are:

   (a) What changes in the ABM specifications need to be made in order to incorporate a distributed version of ECBS?

   (b) What is the performance difference under nominal conditions?

   (c) What is the performance difference when runway reconfigurations take place?

6. **To what extent can the learning-based ECBS mechanism handle aircraft traffic with variable kinematics?**

   The last aspect to address in order to answer the research question is to evaluate the model's behavior under uncertainty. The models built so far have not been evaluated under variable aircraft kinematics. It would be interesting to know whether a learning based system is able to capture variability in aircraft kinematics when making its predictions. Of course, this poses a more difficult learning task in which the generalization performance is expected to degrade. The question is whether this decrease in performance is within a range acceptable for such a system to continue to produce non-conflicting paths for the aircraft. The first step is to define which aspects of the aircraft kinematics will be considered (Table 3.1). Looking at available track data one can derive distributions for these variables and use these as inputs to the simulator. For instance, larger weight category aircraft might have a lower acceleration than medium-sized jets. Such aspects can be reflected on the training data and the generalization performance of the model subsequently be tested. It is not certain whether an analysis with regard to all parameters is feasible as, at this moment, the duration of the system's training phase is not known. Relevant sub-questions are:

   (a) How will variability in aircraft kinematics be incorporated in the model?

   (b) How do the two taxi time prediction mechanisms compare under these circumstances?

   (c) What changes could be made to the architecture or training process of the learned model such that variability can be captured?

6.2. Research Scope

The focus of this MSc thesis is two-fold. First a learning-based cooperative multi-agent planning mechanism will be implemented and its performance will be compared to an existing implementation and to the real-world. The second step involves the extension of the existing cooperative multi-agent planning mechanism to an enhanced version and perform the associated comparisons. The behavior of the mechanism when coupled with the learning mechanism will also be investigated. To assist with the timely execution of the above, the following scope has been defined.

- The research considers a simplified lay-out of Schiphol airport;
- Airport surface movement operations are limited to the taxiway infrastructure;
- Only aircraft movements will be considered;
- Aircraft are assumed to carry out their plans perfectly including no delays;
- Arriving aircraft are assumed to require a route from the point at which they vacated the runway to the apron exit/entry point. Departing aircraft are assumed to require a route from the exit/entry point to the holding point of the active departure runway;
- Aircraft are assumed to follow the Rules of Air as published by ICAO [43];
- The agent-based model specification presented in [36] will be used as a baseline for this study;
- PM and CB highways as presented in [36] will not be considered;

III

Supporting work

A

Model Elaboration

A.1. Modelling assumptions

AAS has the characteristic of being an airport with a complex taxiway system consisting of 6 runways and sees multiple runway reconfigurations during the day. The latter often presents capacity concerns. The airport has to decrease its operational limit to facilitate safe operations because the change in runway directions creates varying, often opposing, traffic flows. This makes AAS an ideal candidate to consider when studying airport surface movement operations, aiming to provide answers to these types of situations. However, some scoping is deemed necessary; otherwise, the problem can quickly increase in complexity. For this reason, several assumptions have been made.

From an operational perspective, only the air-side operations are modelled. That is, from the apron exit until the runway entry and vice versa. Ground vehicles such as shuttles, tow tugs are not considered, and the only users of the runway and taxiway system are the aircraft. This also means that traffic within the apron area will not be modelled. Other types of operations, such as aircraft de-icing and remote holding at parking locations, are also left out of this study. Gates are assumed to always be available for arriving traffic.

In terms of traffic control, arriving aircraft can be controlled as soon as they vacate the runway. It is assumed that they cannot change their speed immediately after runway exit as they are travelling at a higher speed, making it more challenging to slow down or stop. Arriving aircraft are assumed to require a route from the runway exit node to the apron exit/entry point. Departing aircraft are assumed to require a route from the exit/entry apron to the active departure runway's holding point. In reality, these are aircraft that have completed their push back and are ready to taxi. These can remain stationary and move only when they are commanded to do so.

Regarding the aircraft, they all are assumed to have identical properties and are therefore treated similarly. Each aircraft aims at taxiing at its maximum taxi speed v_{max} unless it enters a turn segment larger than a specified threshold, in which case, it slows down to the maximum allowed turn speed v_{turn}. Or when it is in the vicinity of another aircraft. Aircraft have to maintain a minimum separation distance between each other. This distance is set to a constant value (Table A.1) for all aircraft. Next, all aircraft adhere to the commands given by the planning mechanism. Their execution is assumed to occur without any delay. Additionally, aircraft can change their heading and acceleration instantaneously.

Lastly, in terms of runway usage, runway occupancy times have been included. This forbids succeeding aircraft from taking off within a specified time period from each other. Instead, they have to form queues on the taxiway network around the runway holding point while maintaining their separation distance and waiting for their turn to take off. To add to this, no runway schedule is specified in the simulation. Instead, the runway usage depends on the flight's origin-destination pair as defined in the flight schedule used as input to the simulation.

A.2. Relevant agent properties

Agent properties that have been excluded from the Thesis paper are elaborated in this section.

Entry/Exit Agents:

Check Flight Schedule: This property involves an interaction between the Entry Agents and the environment. The Entry Agents check the flight schedule at each time point to determine whether an Aircraft Agent's release time matches the current simulation time point. If so, the Aircraft Agent is added to the Entry Agent's release list. Subsequently, the Entry Agent proceeds with the execution of the *Route Generation and Release* property.

Route Generation and Release: This property involves an interaction between:

– Entry Agents and the environment,

– Entry Agents and Aircraft Agents,

– Entry Agents and ATC Agents

At first, the Entry Agent uses graph information to generate a route for the Aircraft Agent using the A* algorithm. In case of inbound traffic, the Entry Agent coordinates with the ATC agent located at the runway exit to determine the runway exit's availability. If the runway exit is available, the Aircraft Agent is then released. In case of outbound traffic, the Entry Agent coordinated with neighbouring ATC agents to the gate's locations and is informed of any traffic in the adjacent links to the gate. If the neighbourhood around the gate is free, then the Aircraft Agent is released. Otherwise, the release is delayed for a subsequent time point.

Remove Aircraft Agent: This property involves an interaction between ATC and Exit Agents. When an Aircraft Agent reaches its destination, it is handed over from the ATC Agent to the Exit Agent. The Exit Agent then removes it from the simulation environment. Upon doing so, the Exit Agent triggers an occupancy time, during which no other Aircraft Agent can use the relevant runway or make use of a gate.

ATC Agents:

Change Runway Crossing Edge Property: This property defines interactions between the ATC Agents and the Airport Operation Status Agent. The ATC Agent is responsible for closing and re-opening particular runway crossing segments. The information is received by the Airport Operation Status Agent, who keeps track of the Runway Mode of Operations (RMO).

Handover Property: This property defines interactions between ATC Agents. It is executed whenever an aircraft reaches the location of the ATC agent. Once this happens, the Aircraft Agent receives a final set of instructions and the ATC Agent who controls the aircraft hands the responsibility over to the next ATC Agent.

A.3. Simulation Parameters

Table A.1: ATC related simulation parameters.

Symbol	Description	Value
d_{sep}	Minimum separation distance between aircraft	150 m
t_{gate}^{occ}	Gate occupancy time	30 s
t_{rwy}^{occ}	Runway occupancy time	30 s
$t_{rwy,cross}^{occ}$	Runway crossing occupancy time	120 s

Table A.2: Aircraft agent related simulation parameters.

Symbol	Description	Value
v_{max}	Maximum taxi-speed	15.4 m/s
v_{turn}	Maximum taxi-speed in turn segments	5.14 m/s
$acc_{comfort}$	Magnitude of comfort acceleration	0.26 m/s^2
$dec_{comfort}$	Magnitude of comfort deceleration	0.77 m/s^2
dec_{max}	Magnitude of maximum deceleration	5.14 m/s^2
Radar range	Radar range within which other aircraft can be detected	250 m

Table A.3: Other parameters used in the simulations.

Symbol	Description	Value
dt	Timestep in simulator	1 s
$deg_{no,turn}$	Turn degree limit for which no braking is needed	5.14 m/s
$T_{window,CBS}$	CBS conflict time-window	15 s
ATC_{memory}	Duration in which conflicts are stored in ATC Agent's memory	20 s
Uncertainty time window	-	20 s
t_R	Relevance function threshold	0.8

A.4. Forward simulation algorithm

Algorithm 1 Forward Simulation of the Aircraft Agents route

$route \leftarrow$ route of the Aircraft Agent
$n_{index} \leftarrow$ index of current node in route
$n_{total} \leftarrow$ total number of nodes in route
$n_{remain} \leftarrow n_{total} - n_{index}$ number of remaining nodes
$T \leftarrow$ empty list
$D \leftarrow$ empty list
$V_{now} \leftarrow$ velocity of Aircraft Agent at the current time point
$ATC_{app} \leftarrow$ ATC Agent which Aircraft Agent is approaching
$d \leftarrow$ distance to ATC_{app}
$t \leftarrow$ current time point
$wasturn \leftarrow$ None
$t_{delay} \leftarrow 0$
$\theta_{turn} \leftarrow$ turn angle of route segment

if $V > 0$ **then**
    $V \leftarrow V_{now}$
else
    $V \leftarrow 2.6$ m/s
end if

$t_{next} \leftarrow t + d/V$

for *node* **from** n_{index} **until** $n_{remain} - 1$ **do**
    $d \leftarrow d + d_{node}$
    if $wasturn = True$ **then**
        $t_{delay} \leftarrow t_{delay} + \|V_{now} - V_{turn}\|/\alpha_{acc}$
    end if
    $wasturn \leftarrow$ False
    if $\theta_{turn} \geq \theta_{maxturn}$ **and** $V \geq V_{turn}$ **then**
        $t_{delay} \leftarrow t_{delay} + \|V_{turn} - V_{now}\|/\alpha_{dec}$
        $wasturn \leftarrow True$
    end if
    $t_{next} \leftarrow t + d/V_{now} + t_{delay}$
    $T \leftarrow t_{next}$
    $D \leftarrow d$
end for

A.5. Bayesian optimisation algorithm

Algorithm 2 Bayesian optimisation

for n = 1,2,... **do**
    optimise acquisition function a to get $\boldsymbol{x}_{n+1}$
        $\boldsymbol{x}_{n+1} = \text{argmax}_{\boldsymbol{x}}\, a(\boldsymbol{x}; D_n)$
    evaluate objective function to obtain y_{n+1}
    store data $D_{n+1} = D_n, (\boldsymbol{x}_{n+1}, y_{n+1})$
    update statistical model
end for

A.6. Input flight schedule details

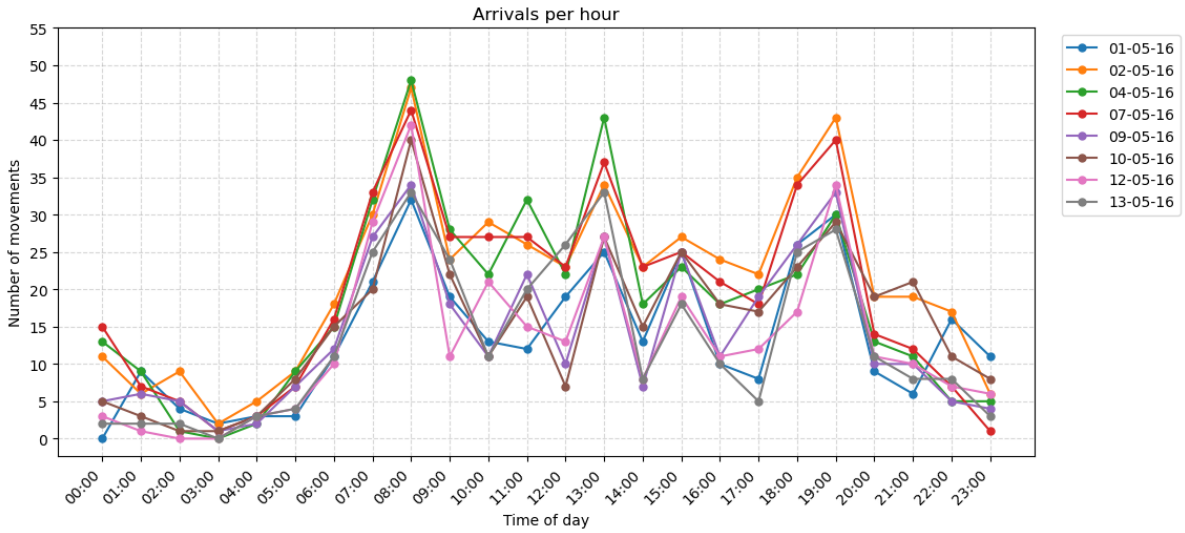

Figure A.1: Arrivals per hour for the 8 days of real world flight schedule.

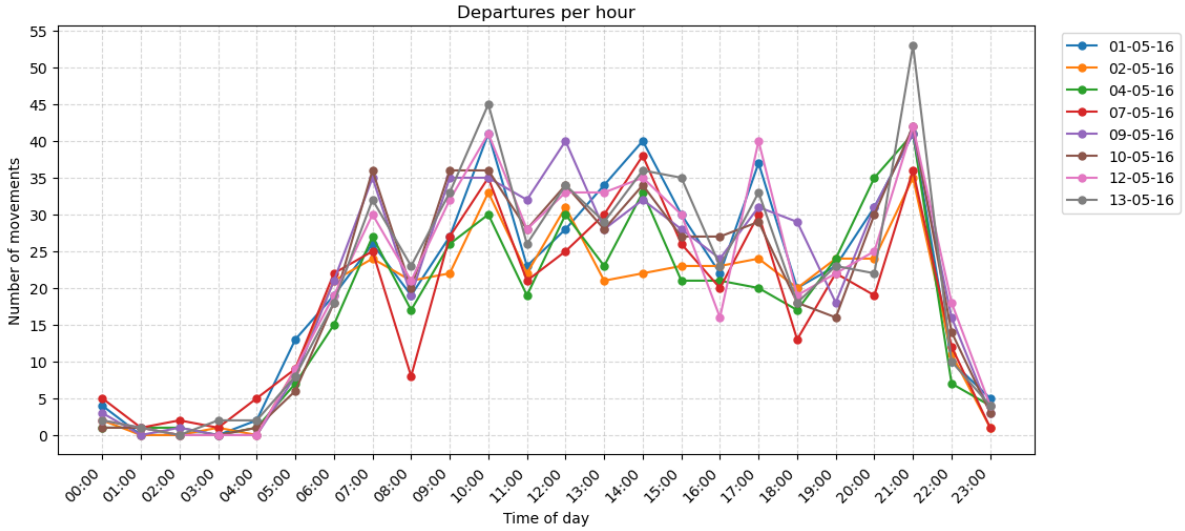

Figure A.2: Departures per hour for the 8 days of real world flight schedule.

A.7. Input flight schedule details for conflict analysis

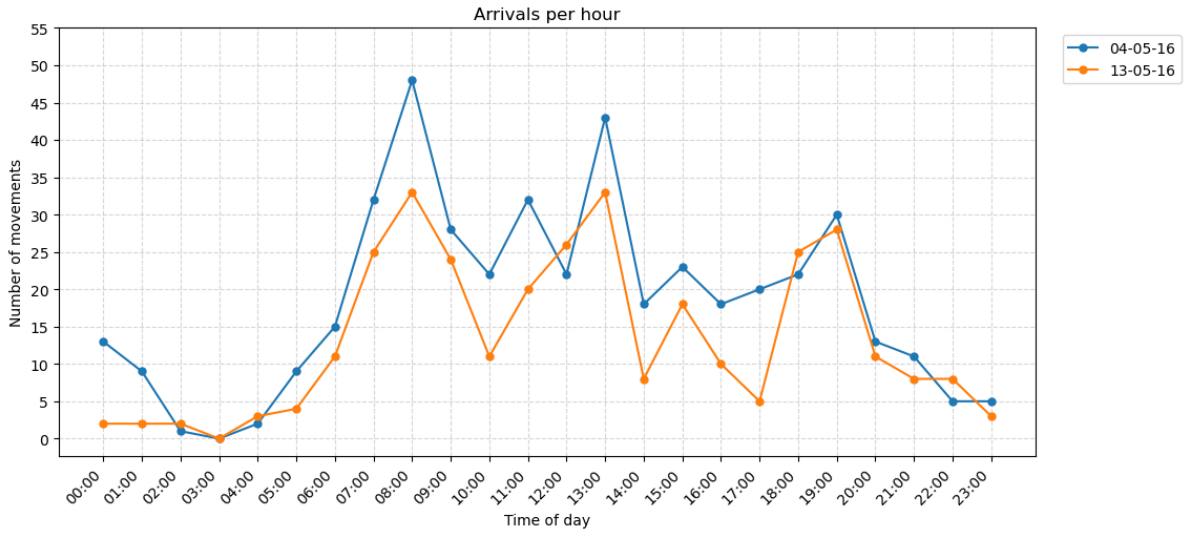

Figure A.3: Arrivals per hour for 4th and 13th of May 2016.

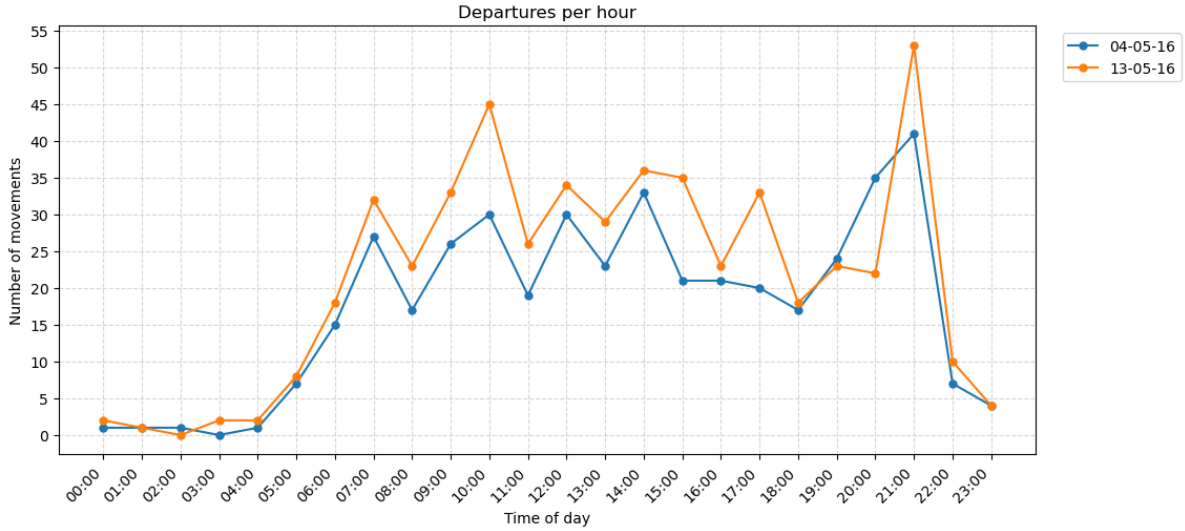

Figure A.4: Departures per hour for 4th and 13th of May 2016.

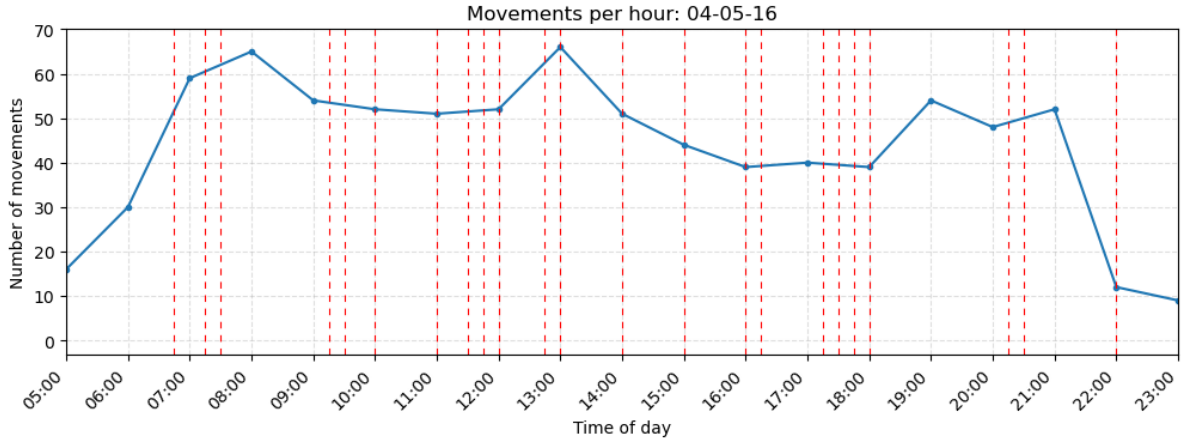

Figure A.5: Number of movements (blue) and runway configuration occurrences (red) for 04-05-16.

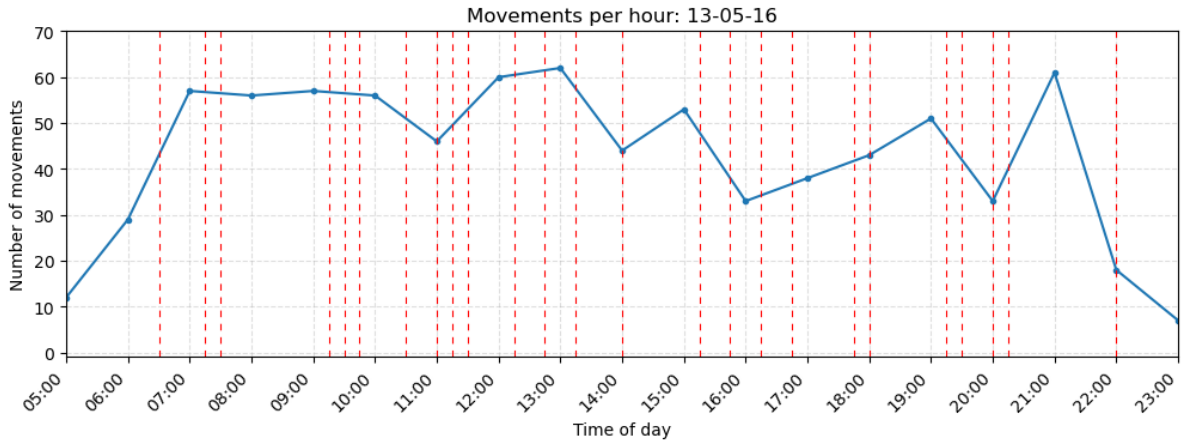

Figure A.6: Number of movements (blue) and runway configuration occurrences (red) for 13-05-16.

B

Simulation results

B.1. Taxi-time results for individual days

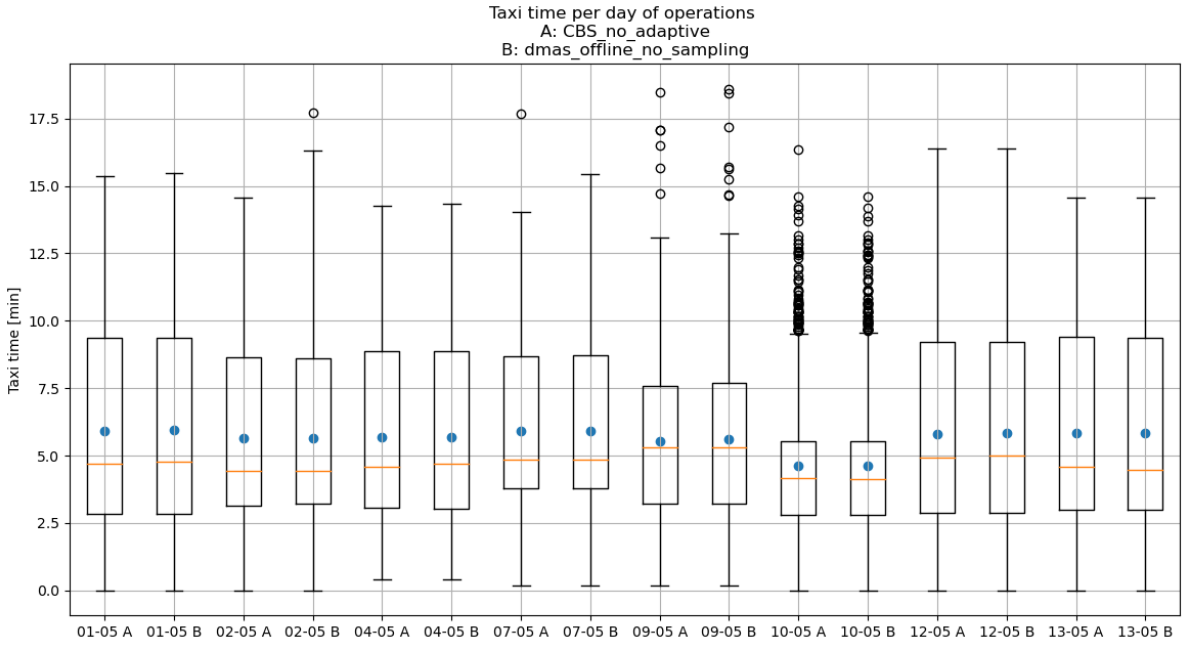

Figure B.1: Taxi-time distributions per flight day between Baseline and dMAS CBS.

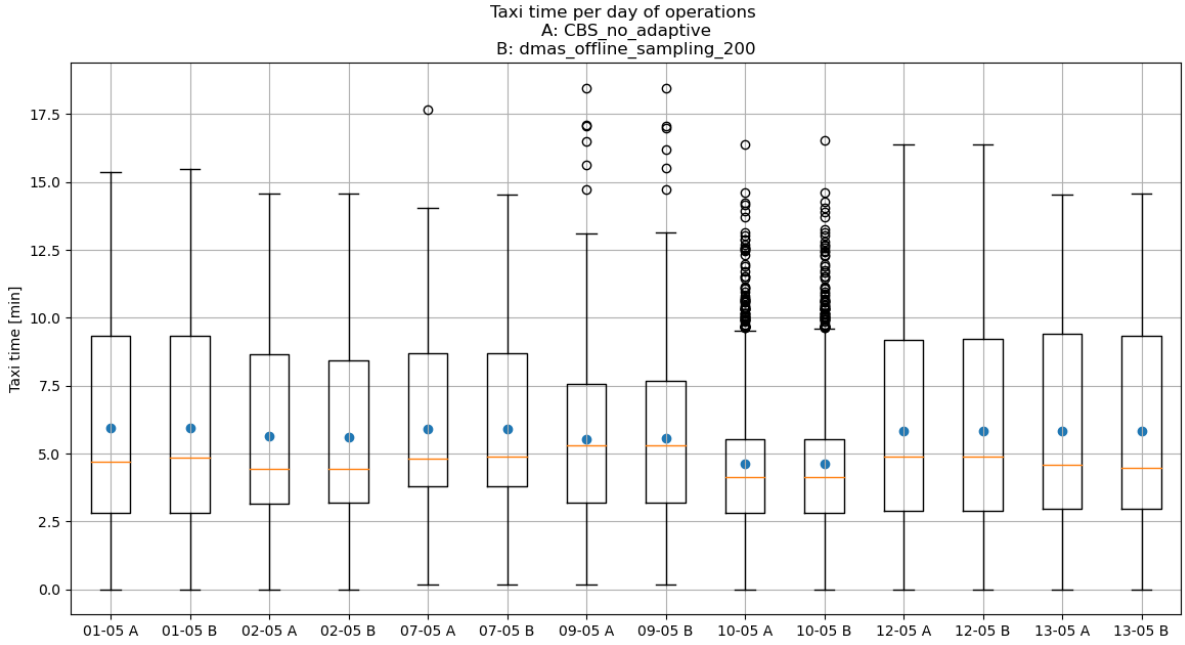

Figure B.2: Taxi-time distributions per flight day between Baseline and dMAS-U(200) CBS.

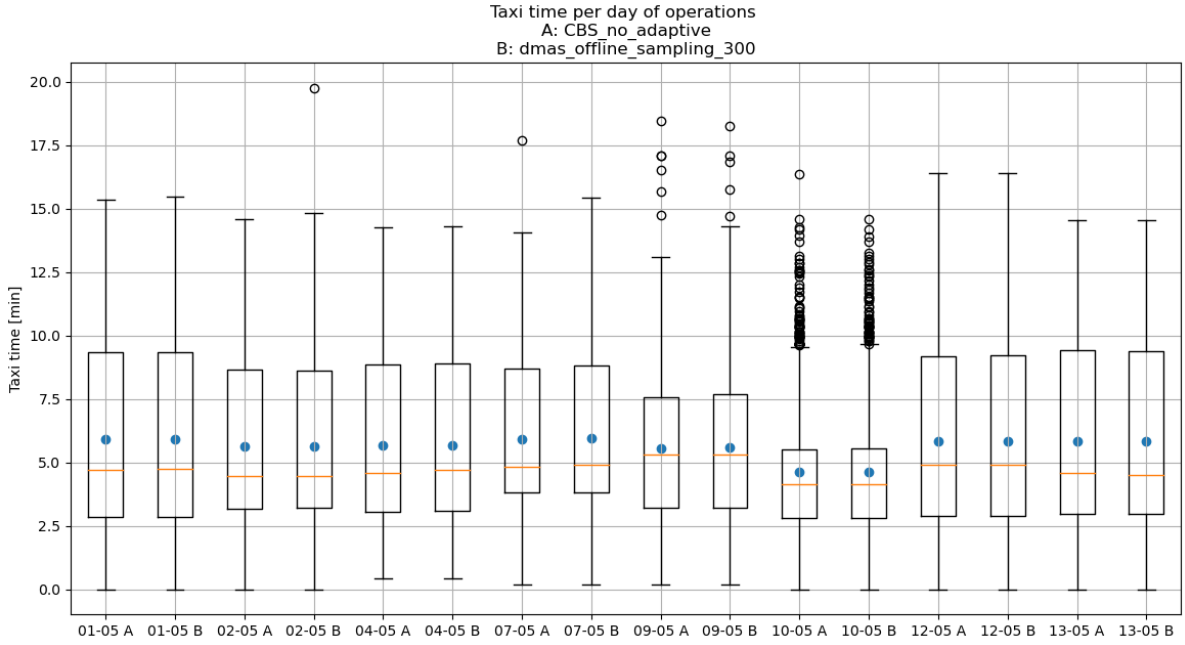

Figure B.3: Taxi-time distributions per flight day between Baseline and dMAS-U(300) CBS.

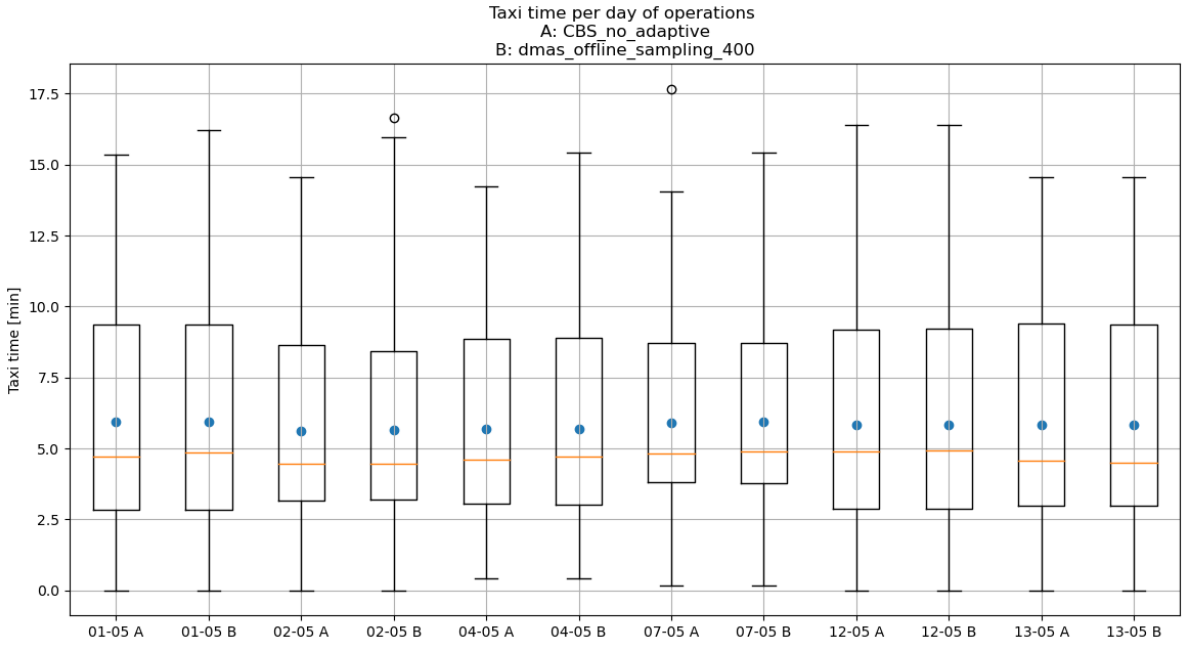

Figure B.4: Taxi-time distributions per flight day between Baseline and dMAS-U(400) CBS.

B.2. Taxi-distance results for individual days

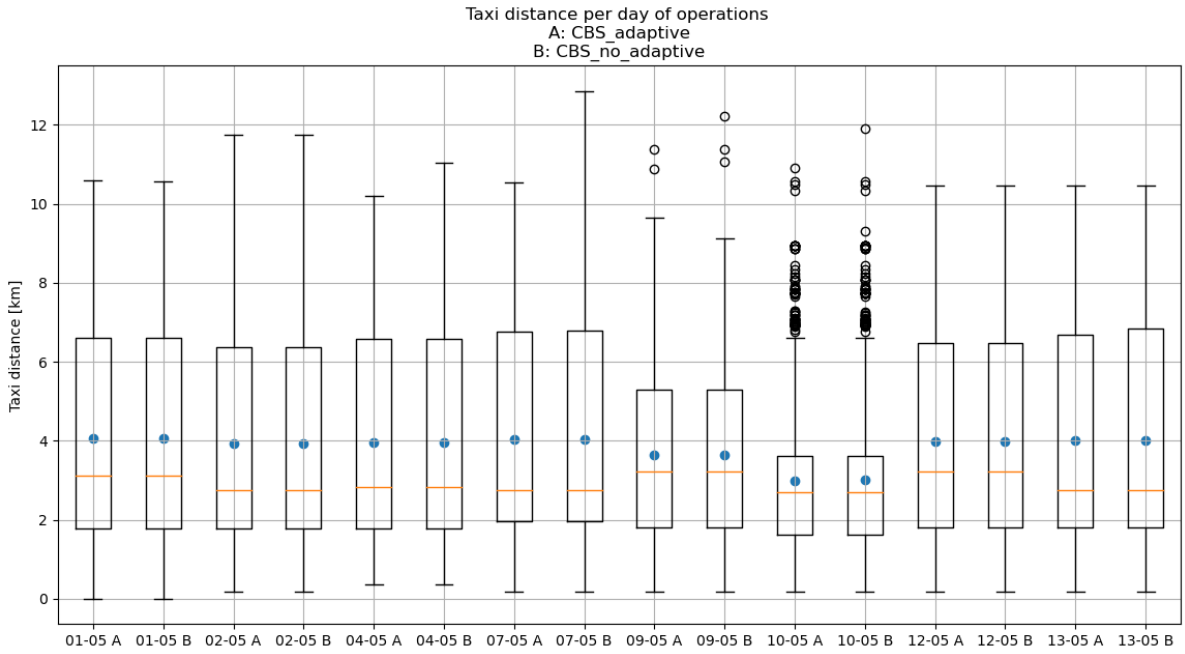

Figure B.5: Taxi-distance distributions per flight day between Baseline and dMAS CBS.

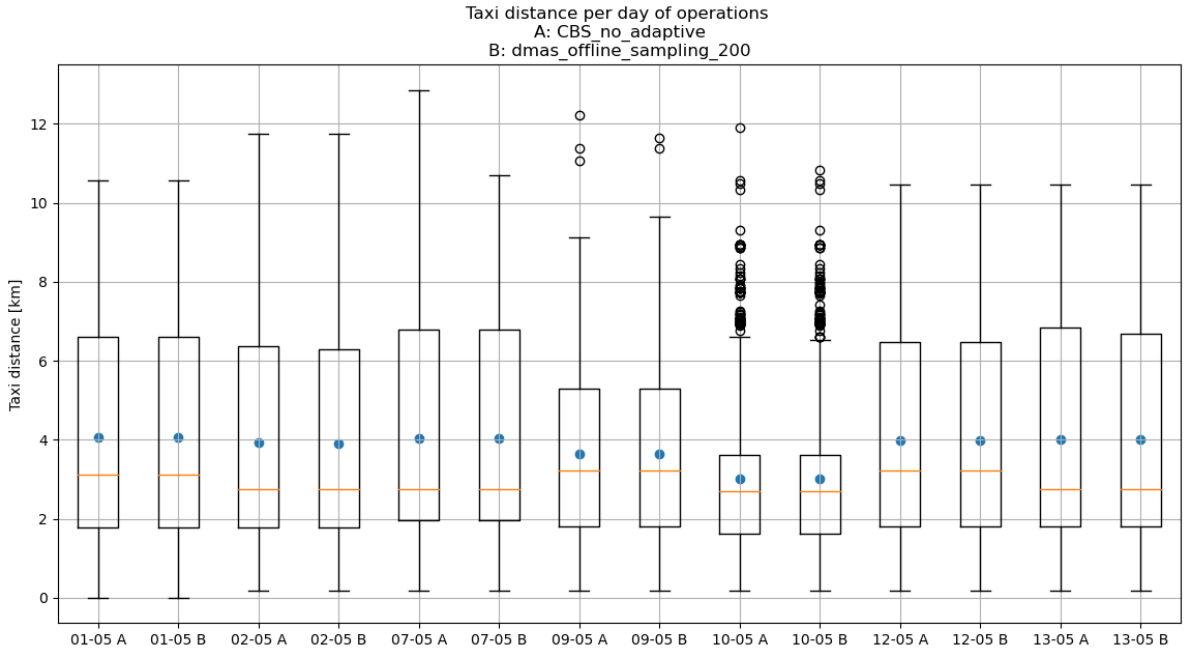

Figure B.6: Taxi-distance distributions per flight day between Baseline and dMAS-U(200) CBS.

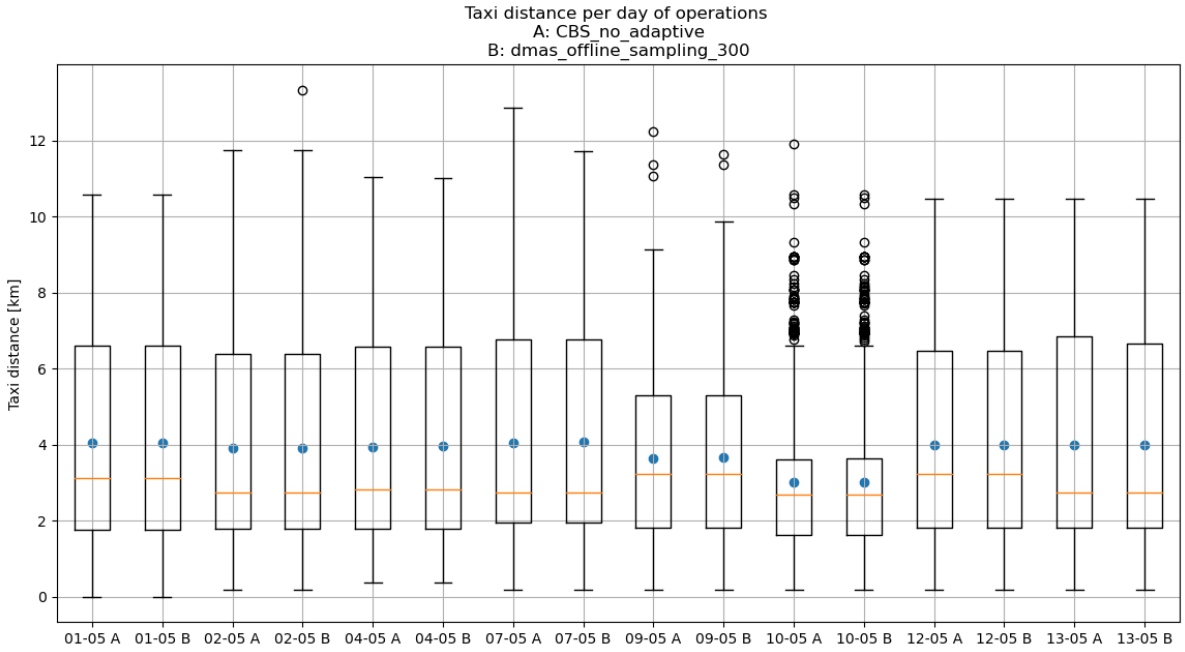

Figure B.7: Taxi-distance distributions per flight day between Baseline and dMAS-U(300) CBS.

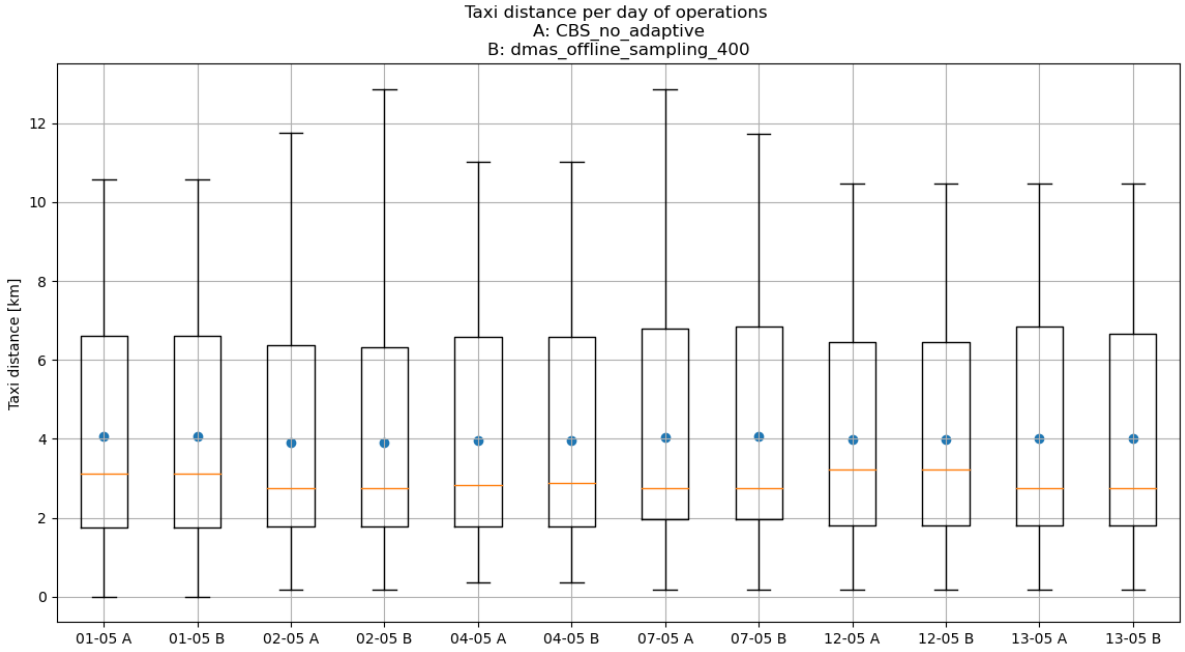

Figure B.8: Taxi-distance distributions per flight day between Baseline and dMAS-U(400) CBS.

B.3. Taxi-speed results for individual days

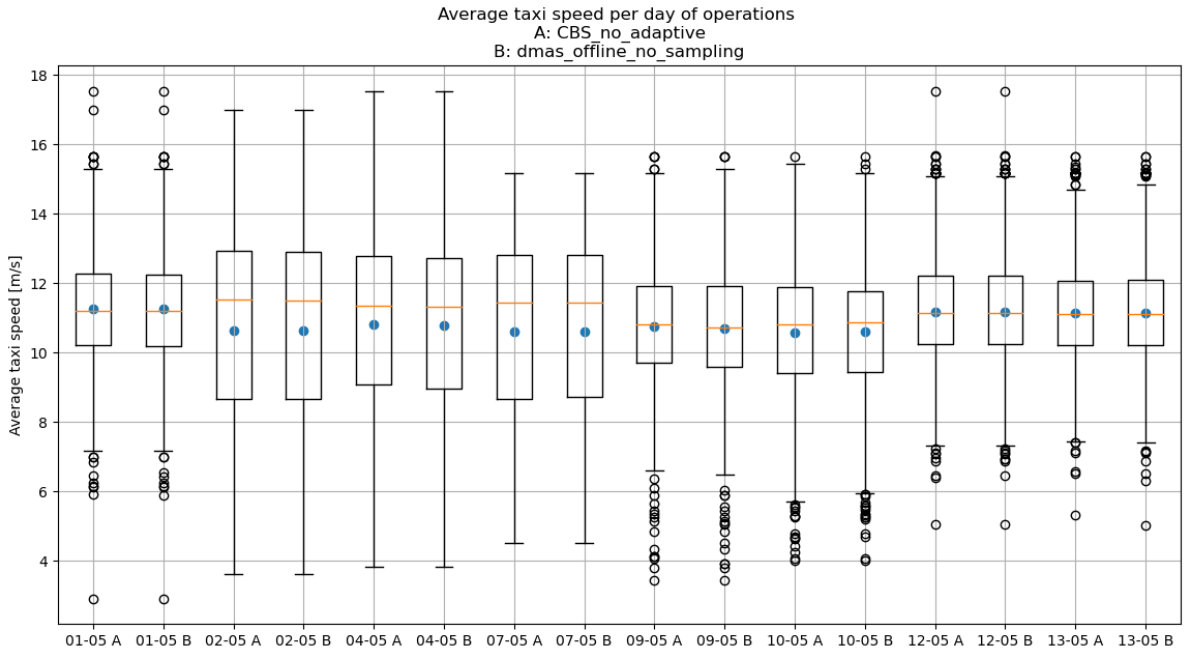

Figure B.9: Taxi-speed distributions per flight day between Baseline and dMAS CBS.

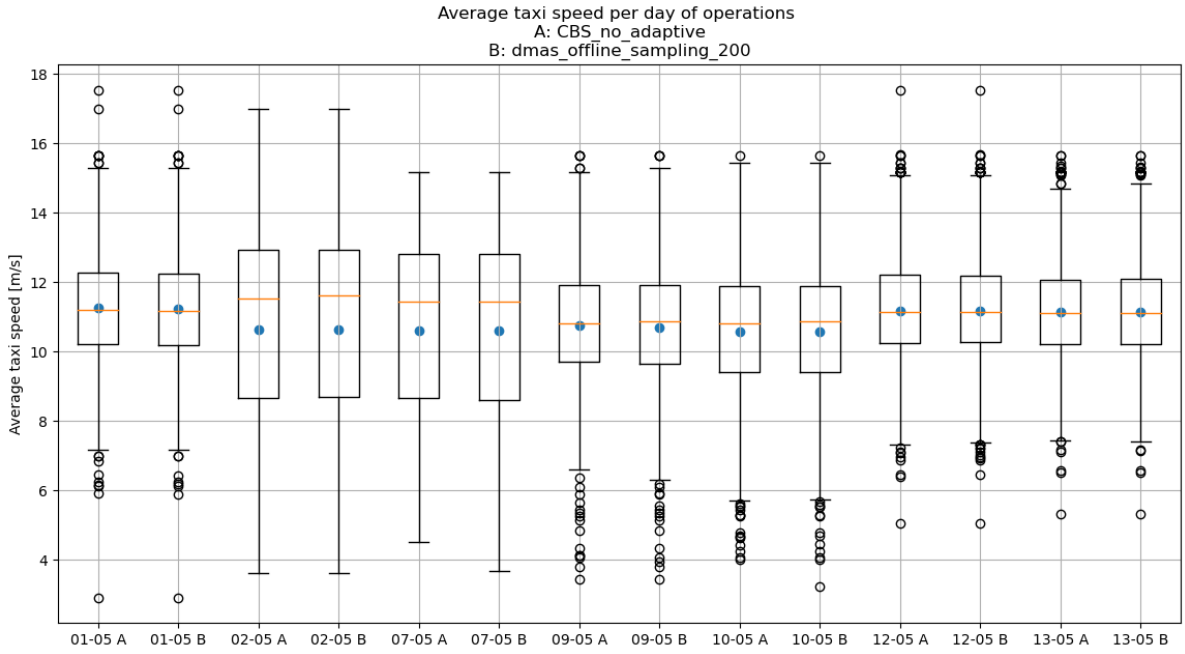

Figure B.10: Taxi-speed distributions per flight day between Baseline and dMAS-U(200) CBS.

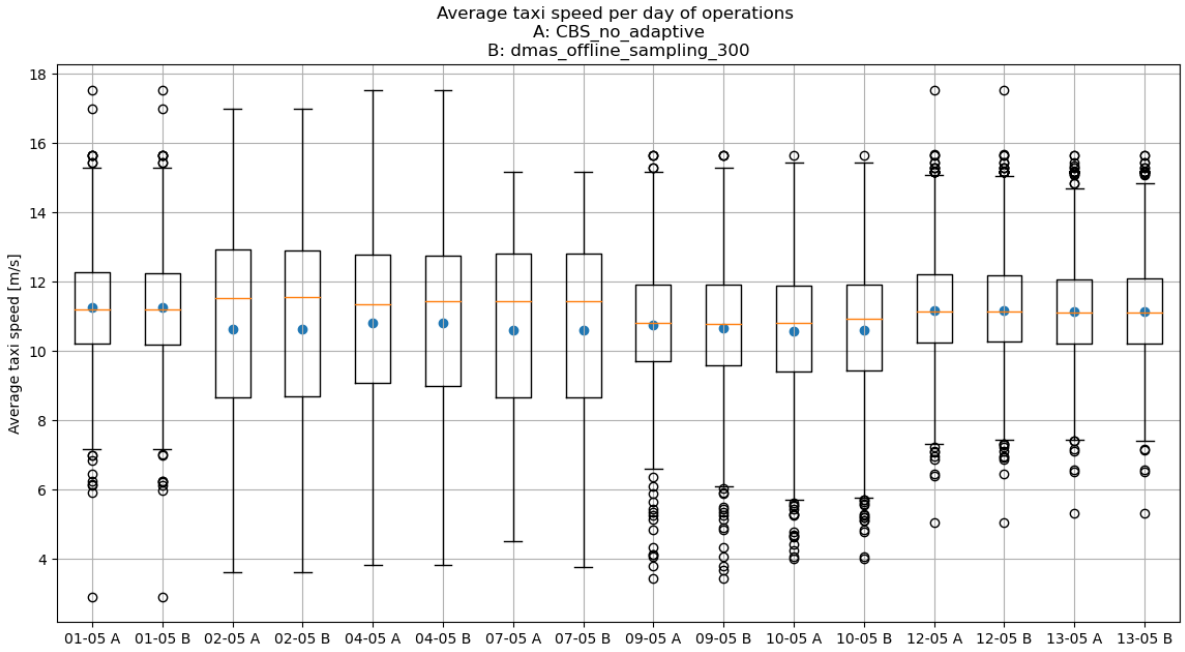

Figure B.11: Taxi-speed distributions per flight day between Baseline and dMAS-U(300) CBS.

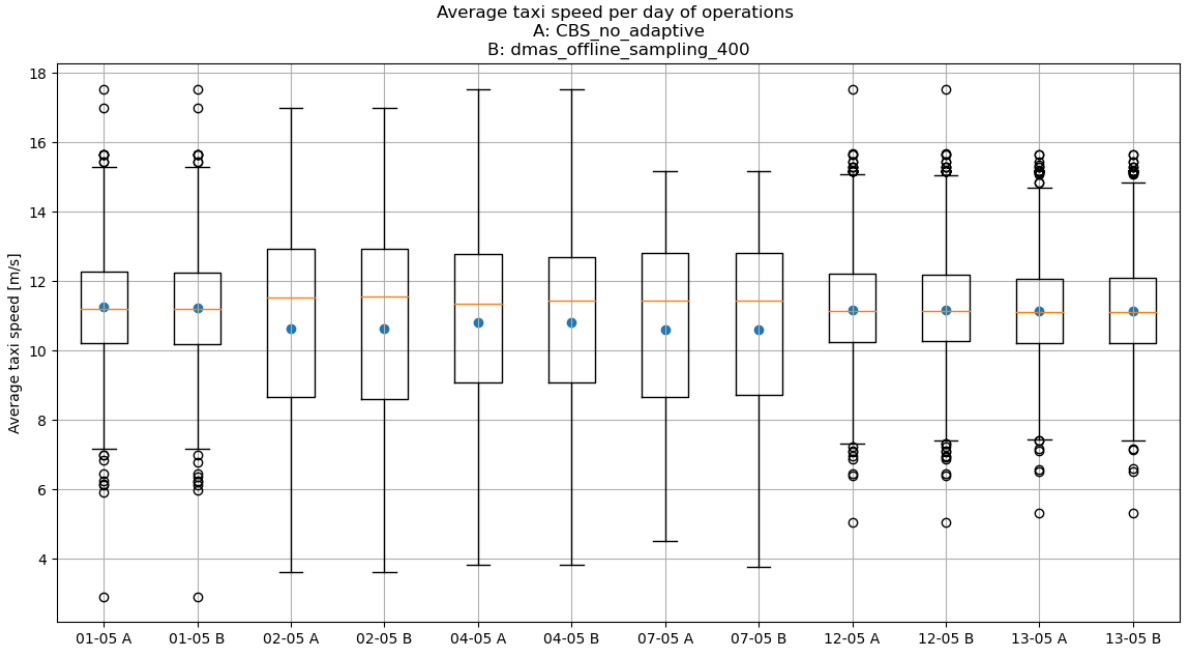

Figure B.12: Taxi-speed distributions per flight day between Baseline and dMAS-U(400) CBS.

B.4. Link participation per sampling strategy

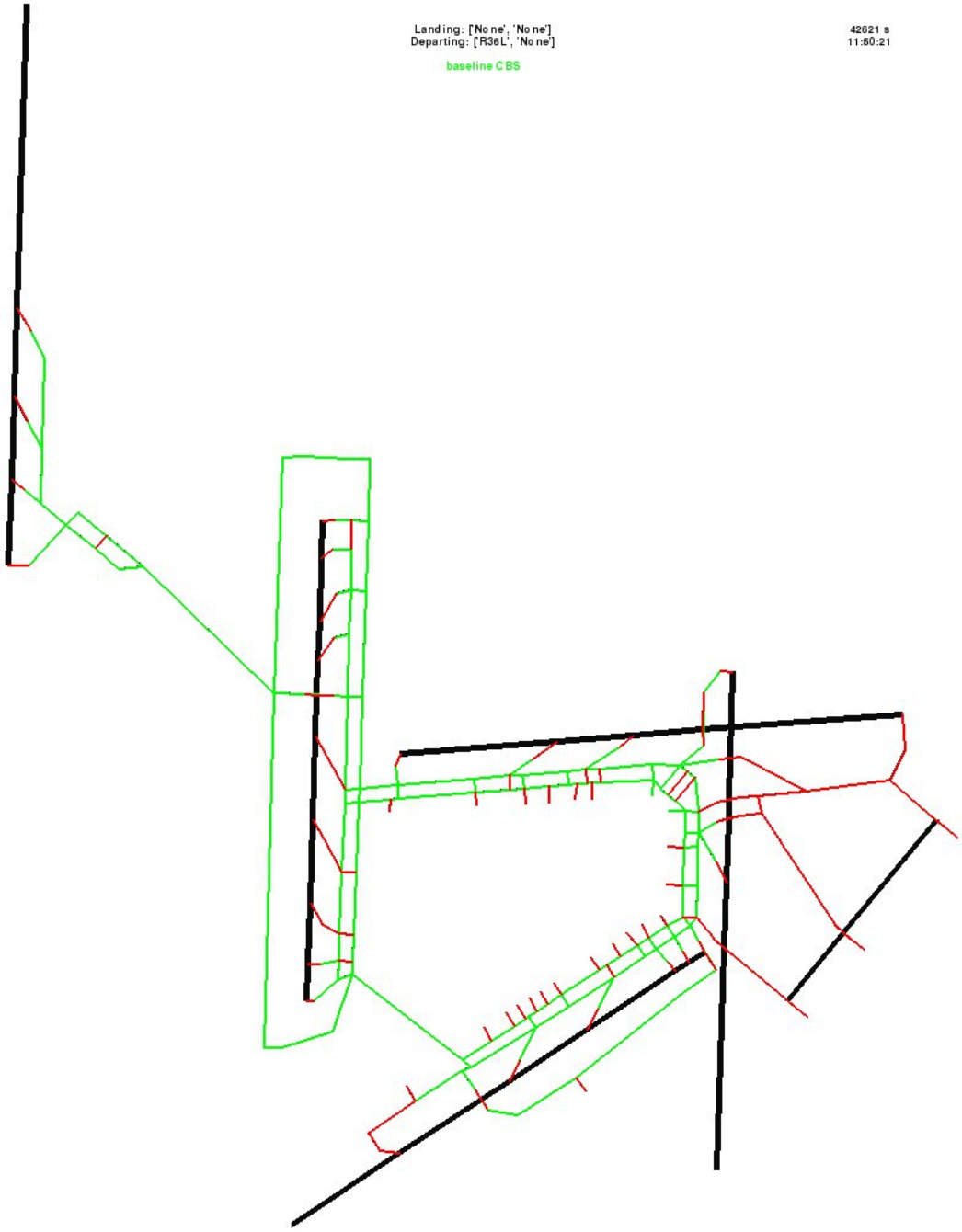

Figure B.13: Link participation when no undersampling was applied.

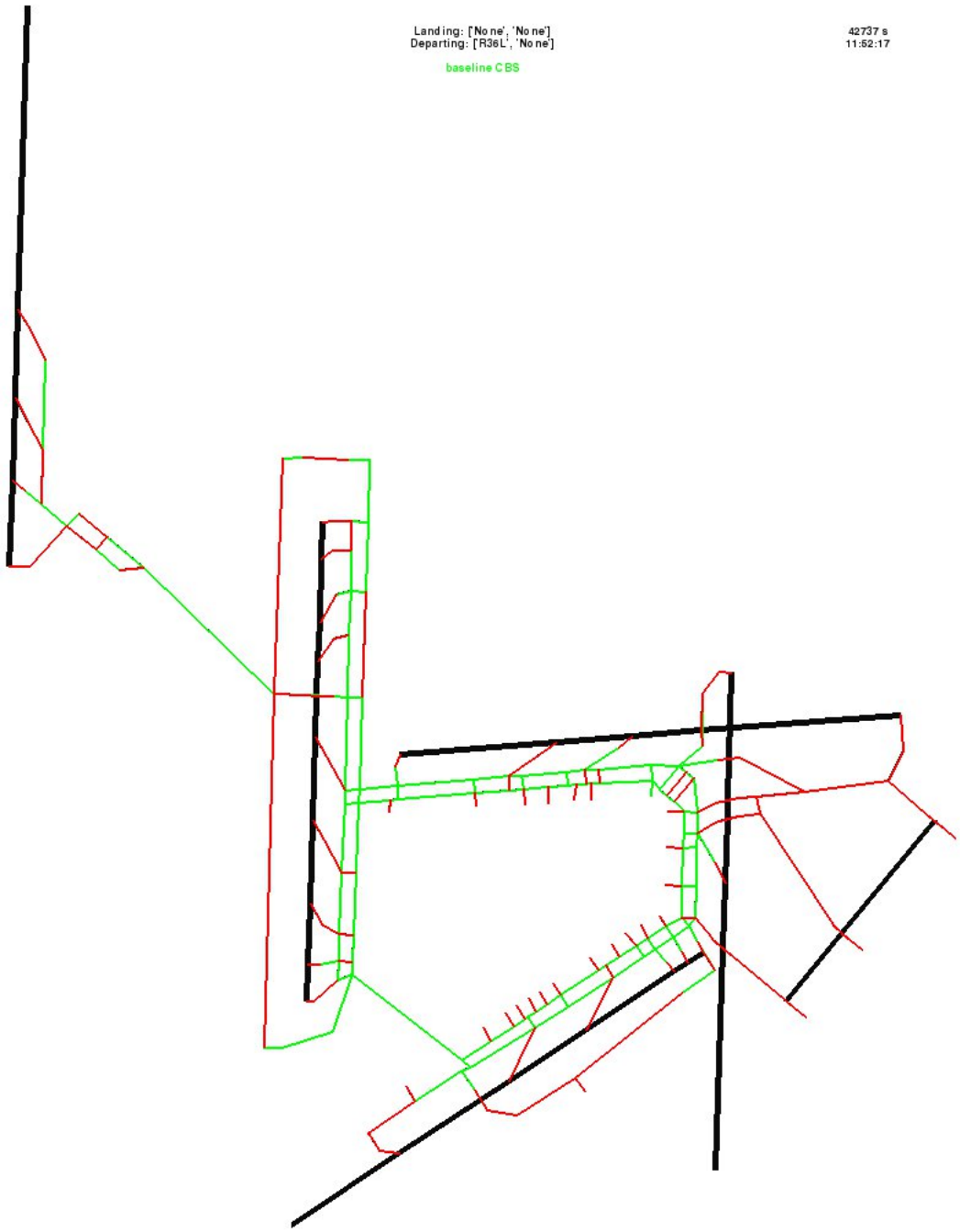

Figure B.14: Link participation when undersampling ratio of 200 was applied.

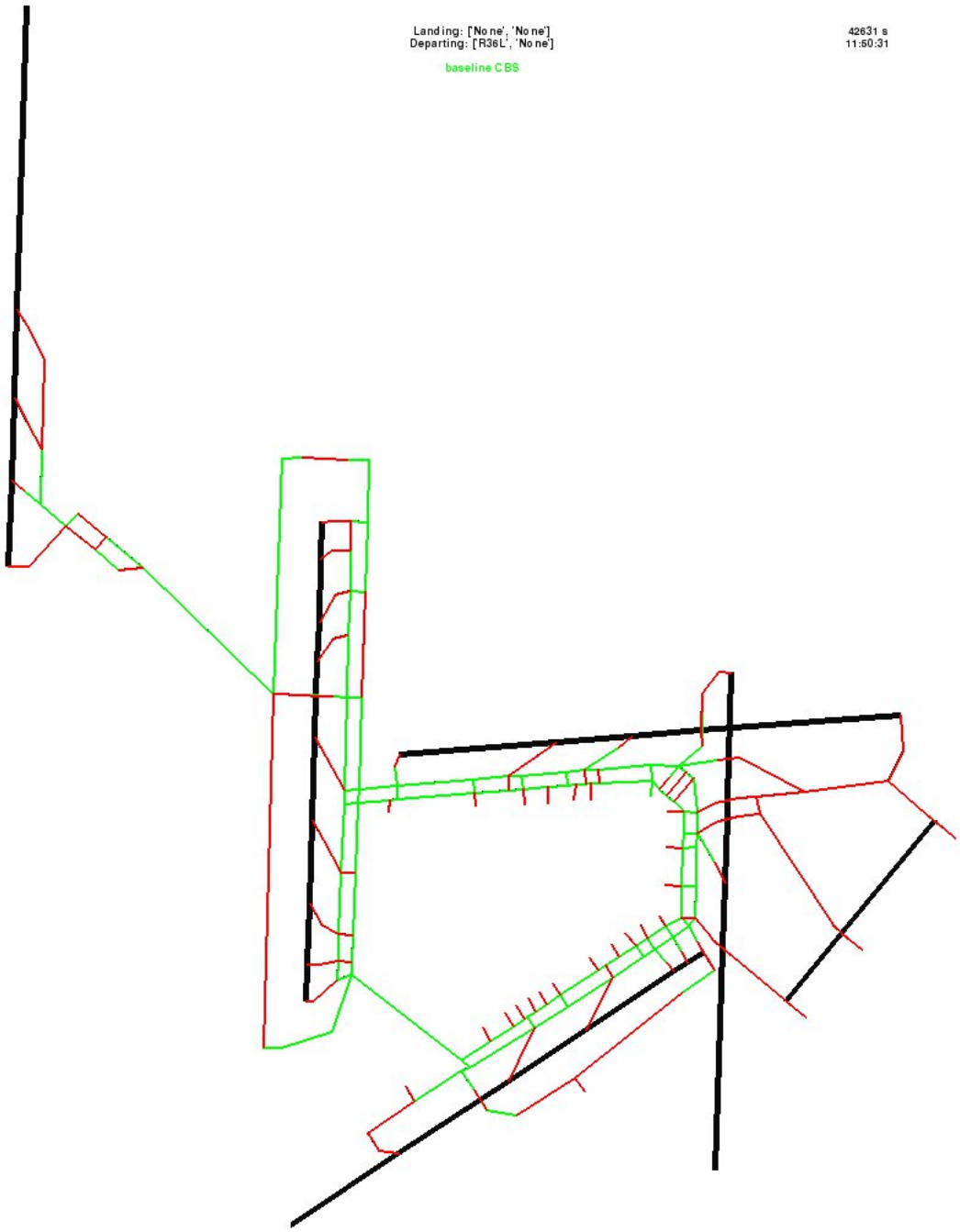

Figure B.15: Link participation when undersampling ratio of 300 was applied.

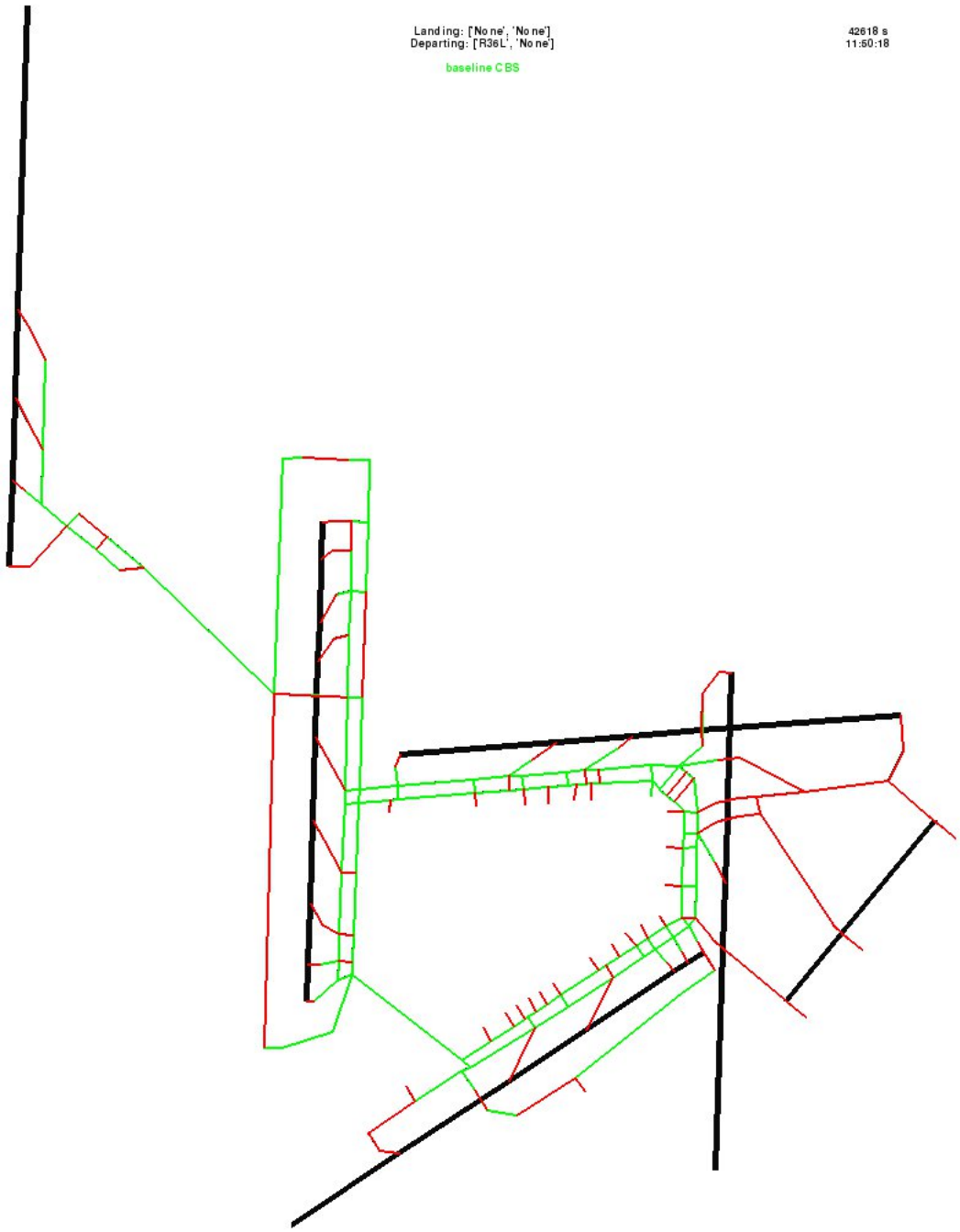

Figure B.16: Link participation when undersampling ratio of 400 was applied.

B.5. Absolute prediction error for Scenarios B and D

CBS Variant	$N_{conflicts}$	$AC_{c_{AE}}$ [s]	$AC_{o_{AE}}$ [s]	CBS_{AE} [s]
Adaptive CBS	2808	$\mu = 24.98\ \sigma = 53.73$	$\mu = 45.36\ \sigma = 65.82$	$\mu = 34.48\ \sigma = 35.66$
Baseline CBS	117	$\mu = 47.66\ \sigma = 50.46$	$\mu = 77.34\ \sigma = 60.16$	$\mu = 59.05\ \sigma = 56.14$
dMAS-CBS	108	$\mu = 51.92\ \sigma = 34.76$	$\mu = 72.44\ \sigma = 57.42$	$\mu = 31.38\ \sigma = 40.98$
U(200)	97	$\mu = 67.84\ \sigma = 37.70$	$\mu = 69.93\ \sigma = 33.36$	$\mu = 33.39\ \sigma = 25.16$
U(300)	79	$\mu = 64.77\ \sigma = 38.30$	$\mu = 74.13\ \sigma = 36.16$	$\mu = 39.47\ \sigma = 30.37$
U(400)	105	$\mu = 55.44\ \sigma = 30.32$	$\mu = 68.23\ \sigma = 34.33$	$\mu = 37.52\ \sigma = 28.10$

Table B.1: Statistics of the absolute prediction error for prediction made in Scenario B.

CBS Variant	$N_{conflicts}$	$AC_{c_{AE}}$ [s]	$AC_{o_{AE}}$ [s]	CBS_{AE} [s]
Adaptive CBS	781	$\mu = 23.96\ \sigma = 39.69$	$\mu = 34.09\ \sigma = 47.17$	$\mu = 32.79\ \sigma = 32.14$
Baseline CBS	29	$\mu = 50.21\ \sigma = 56.39$	$\mu = 59.14\ \sigma = 76.91$	$\mu = 50.66\ \sigma = 40.28$
dMAS-CBS	36	$\mu = 41.86\ \sigma = 32.04$	$\mu = 62.33\ \sigma = 43.85$	$\mu = 33.25\ \sigma = 24.69$
U(200)	17	$\mu = 38.35\ \sigma = 26.93$	$\mu = 59.24\ \sigma = 25.45$	$\mu = 30.65\ \sigma = 29.07$
U(300)	26	$\mu = 43.77\ \sigma = 28.90$	$\mu = 72.65\ \sigma = 42.92$	$\mu = 36.15\ \sigma = 25.82$
U(400)	21	$\mu = 38.29\ \sigma = 24.34$	$\mu = 66.57\ \sigma = 34.69$	$\mu = 42.57\ \sigma = 39.58$

Table B.2: Statistics of the absolute prediction error for prediction made in Scenario D.

B.6. Absolute error distributions for all mechanisms and scenarios

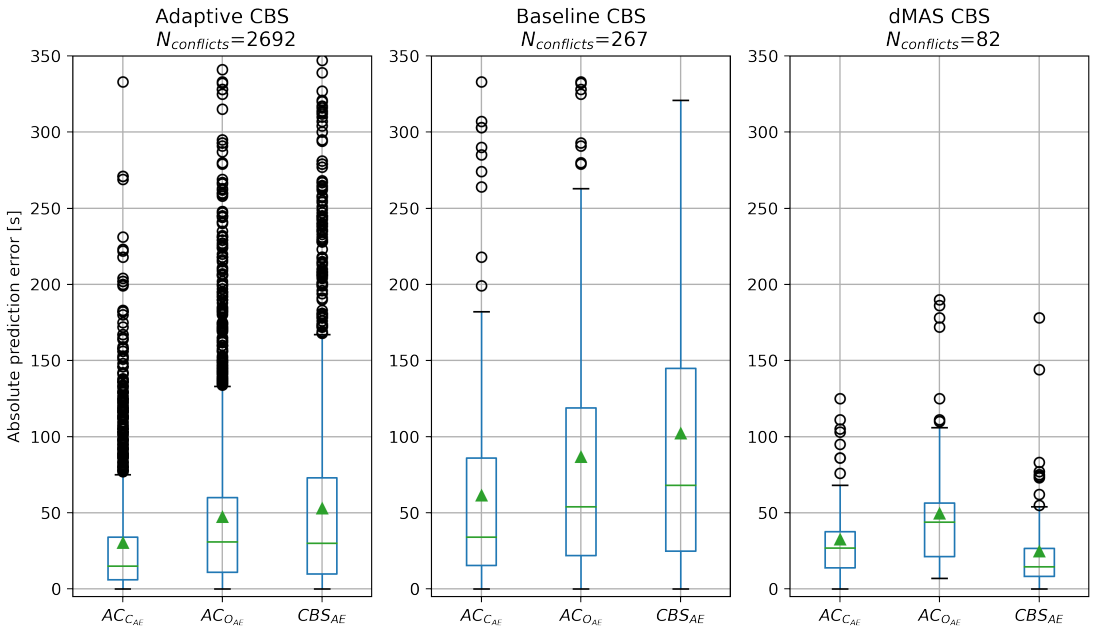

Figure B.17: Conflict prediction error distributions of Adaptive, Baseline and dMAS CBS in Scenario A.

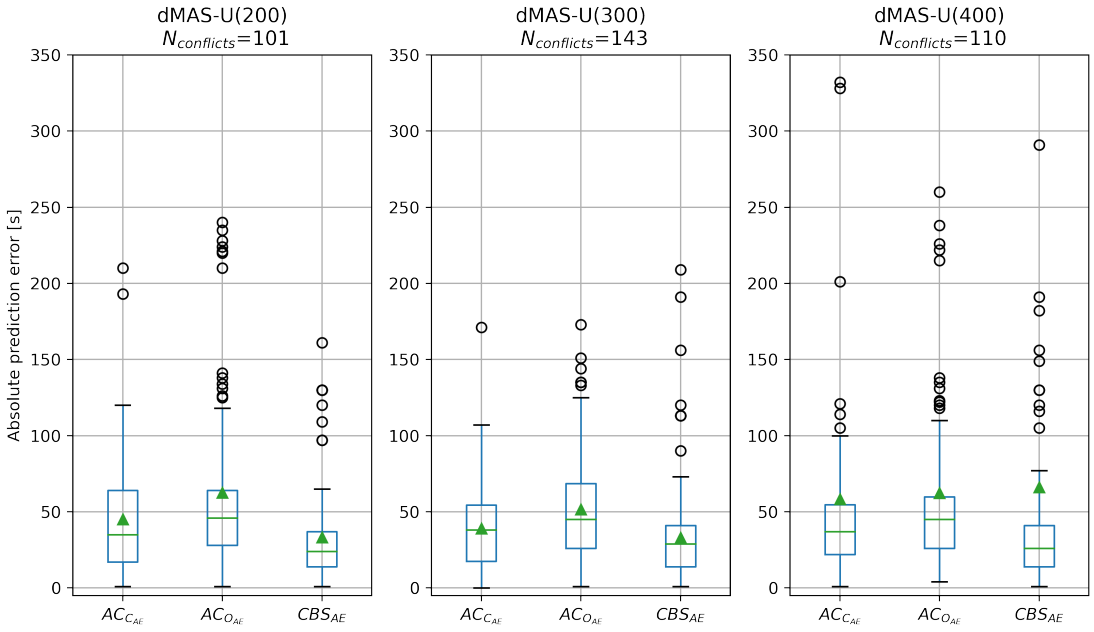

Figure B.18: Conflict prediction error distributions of dMAS-U(200), dMAS-U(300) and dMAS-U(400) CBS in Scenario A.

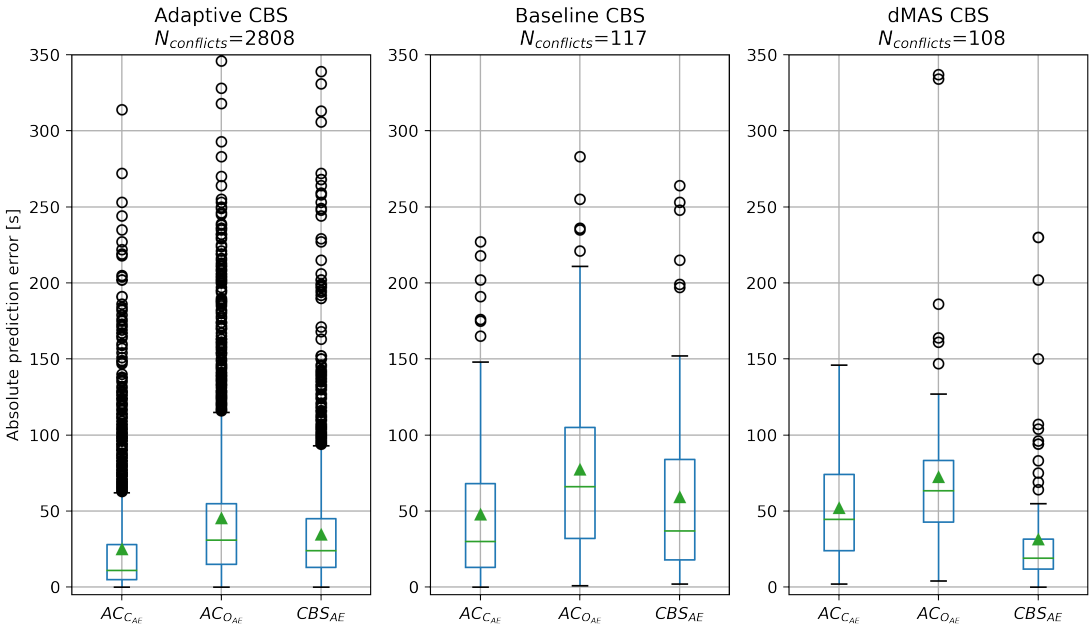

Figure B.19: Conflict prediction error distributions of Adaptive, Baseline and dMAS CBS in Scenario B.

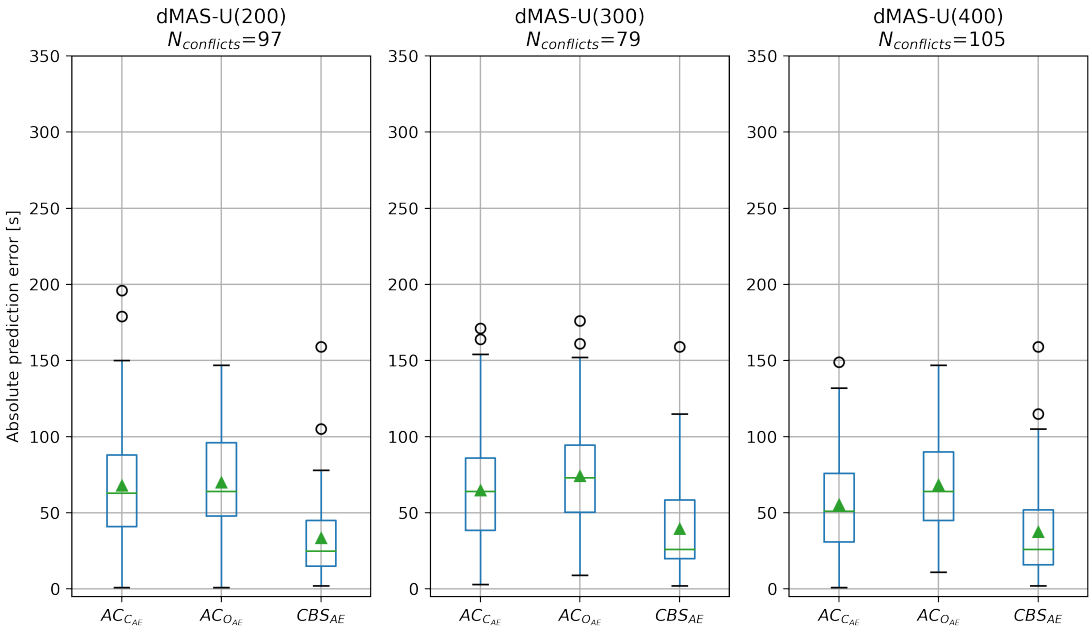

Figure B.20: Conflict prediction error distributions of dMAS-U(200), dMAS-U(300) and dMAS-U(400) CBS in Scenario B.

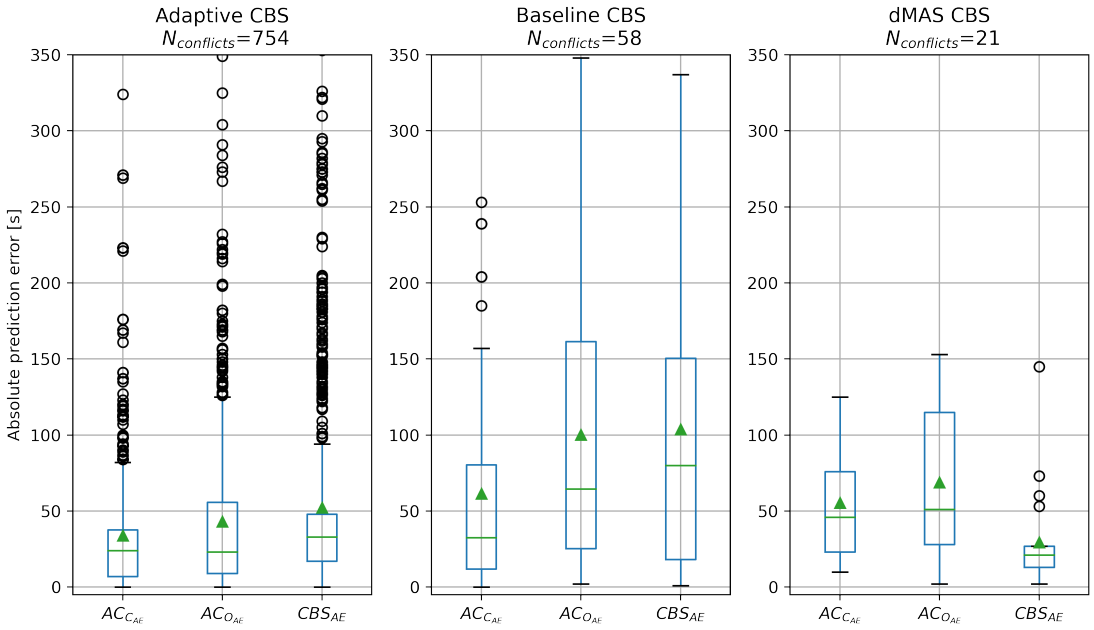

Figure B.21: Conflict prediction error distributions of Adaptive, Baseline and dMAS CBS in Scenario C.

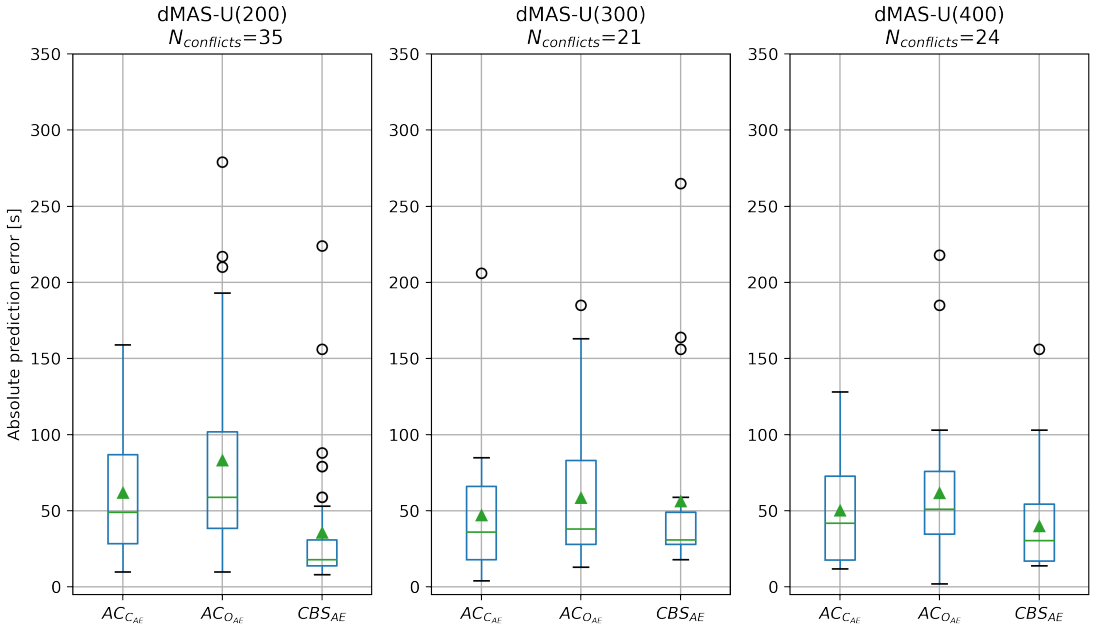

Figure B.22: Conflict prediction error distributions of dMAS-U(200), dMAS-U(300) and dMAS-U(400) CBS in Scenario C.

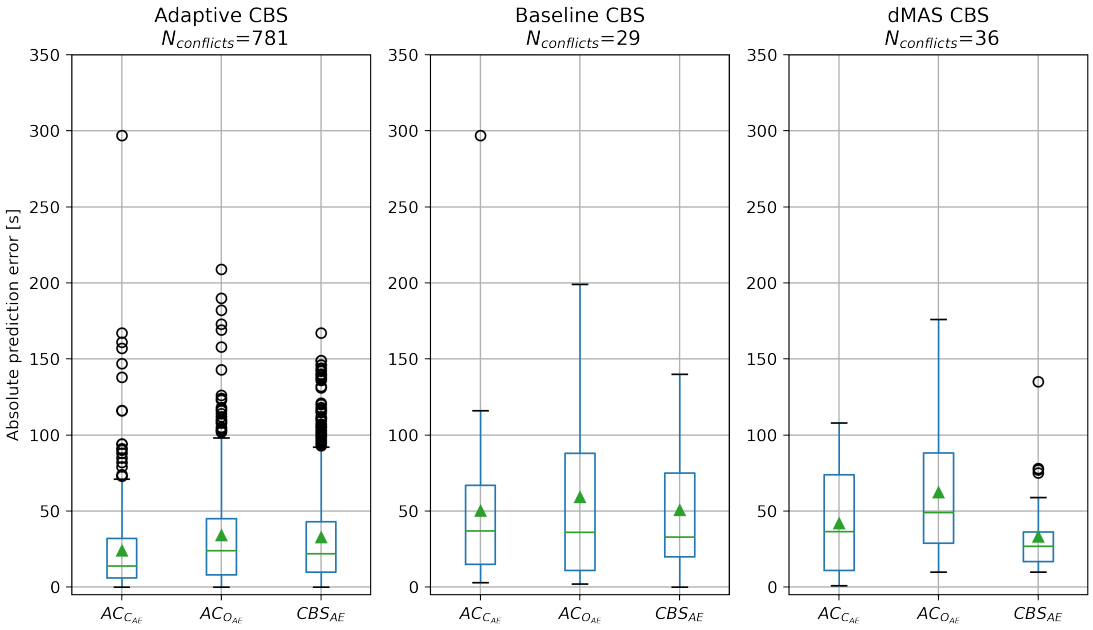

Figure B.23: Conflict prediction error distributions of Adaptive, Baseline and dMAS CBS in Scenario D.

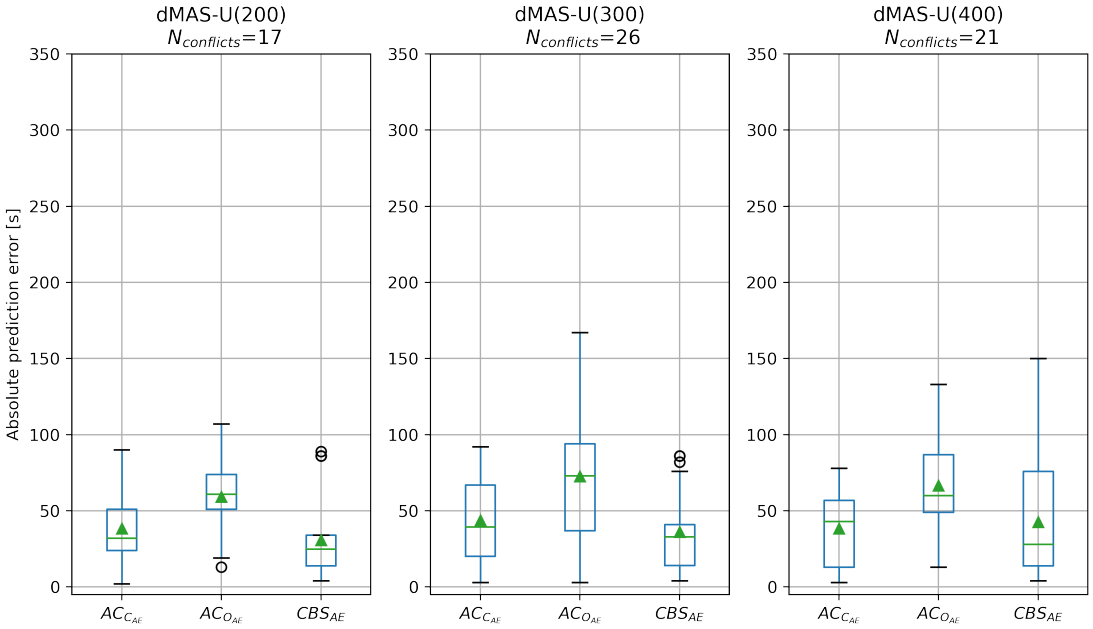

Figure B.24: Conflict prediction error distributions of dMAS-U(200), dMAS-U(300) and dMAS-U(400) CBS in Scenario D.

Bibliography

[1] Airports Council International (ACI). Preliminary world airport traffic rankings released. URL https://aci.aero/news/2019/03/13/preliminary-world-airport-traffic-rankings-released/. Accessed: 14-01-2020.

[2] J. A.D. Atkin, E. K. Burke, and S. Ravizza. The airport ground movement problem: Past and current research and future directions. *Research in Air Transportation*, pages 131–138, 2010. doi: 10.1.221.4402.

[3] JAD Atkin, EK Burke, and S Ravizza. A Statistical Approach for Taxi Time Estimation at London Heathrow Airport. *Cs.Nott.Ac.Uk*, pages 61–63, 2011. URL http://www.cs.nott.ac.uk/{~}smr/share/11{_}MAPSP{_}Ravizza.pdf.

[4] Poornima Balakrishna, Rajesh Ganesan, and Lance Sherry. Application of reinforcement learning algorithms for predicting taxi-out times. *Proceedings of the 8th USA/Europe Air Traffic Management Research and Development Seminar, ATM 2009*, pages 255–261, 2009.

[5] Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. *Proceedings of the 7th Annual Symposium on Combinatorial Search, SoCS 2014*, 2014-Janua(SoCS):19–27, 2014.

[6] Anthony Barrett and Daniel S Weld. Partial-order planning: evaluating possible efficiency gains. *Artificial Intelligence*, 67(1):71–112, 1994.

[7] Zahy Bnaya and Ariel Felner. Conflict-oriented windowed hierarchical cooperative a*. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3743–3748. IEEE, 2014.

[8] Adi Botea, Davide Bonusi, and Pavel Surynek. Solving multi-agent path finding on strongly biconnected digraphs. *Journal of Artificial Intelligence Research*, 62:273–314, 2018.

[9] Eli Boyarski, Ariel Felner, Roni Stern, Guni Sharon, David Tolpin, Oded Betzalel, and Eyal Shimony. ICBS: Improved conflict-based search algorithm for multi-agent pathfinding. *IJCAI International Joint Conference on Artificial Intelligence*, 2015-Janua(Ijcai):740–746, 2015. ISSN 10450823.

[10] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[11] Lucian Buoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. *Studies in Computational Intelligence*, 310:183–221, 2010. ISSN 1860949X. doi: 10.1007/978-3-642-14435-6_7.

[12] Michal Cáp, Peter Novák, Jirí Vokrínek, and Michal Pechoucek. Multi-agent RRT*: Sampling-based Cooperative Pathfinding. *CoRR*, abs/1302.2, 2013. URL http://arxiv.org/abs/1302.2828.

[13] Jun Chen, Stefan Ravizza, Jason A.D. Atkin, and Paul Stewart. On the utilisation of fuzzy rule-based systems for taxi time estimations at Airports. *OpenAccess Series in Informatics*, 20:134–145, 2011. ISSN 21906807. doi: 10.4230/OASIcs.ATMOS.2011.134.

[14] Wei Chen and Keith S Decker. Managing Multi-Agent Coordination , Planning , and Scheduling. 2004.

[15] Satyendra Chouhan and Rajdeeep Niyogi. DMAPP: A Distributed Multi-Agent Path Planning Algorithm. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9457(December), 2015. ISSN 16113349. doi: 10.1007/978-3-319-26350-2.

[16] Satyendra Singh Chouhan and Rajdeep Niyogi. DiMPP: a complete distributed algorithm for multi-agent path planning. *Journal of Experimental and Theoretical Artificial Intelligence*, 29(6):1129–1148, 2017. ISSN 13623079. doi: 10.1080/0952813X.2017.1310142. URL http://dx.doi.org/10.1080/0952813X.2017.1310142.

[17] Rutger Claes. Anticipatory vehicle routing, 2015. URL $https://lirias.kuleuven.be/retrieve/319948thesis.pdf$[freelyavailable].

[18] Liron Cohen and Sven Koenig. Bounded suboptimal multi-agent path finding using highways. *IJCAI International Joint Conference on Artificial Intelligence*, 2016-Janua:3978–3979, 2016. ISSN 10450823.

[19] Liron Cohen, Tansel Uras, and Sven Koenig. Feasibility Study: Using Highways for Bounded-Suboptimal Multi-Agent Path Finding. *Proceedings of the 8th Annual Symposium on Combinatorial Search, SoCS 2015*, 2015-Janua:2–8, 2015.

[20] Liron Cohen, Tansel Uras, T. K. Satish Kumar, Hong Xu, Nora Ayanian, and Sven Koenig. Improved solvers for bounded-suboptimal multi-agent path finding. *IJCAI International Joint Conference on Artificial Intelligence*, 2016-Janua:3067–3074, 2016. ISSN 10450823.

[21] Daniel D. Corkill. Hierarchical planning in a distributed environment. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI79, page 168175, San Francisco, CA, USA, 1979. Morgan Kaufmann Publishers Inc. ISBN 0934613478.

[22] W.H. Dalmeijer. Schiphol gebruiksprognose 2020, 2019.

[23] Boris de Wilde, Adriaan W. ter Mors, and Cees Witteveen. Push and rotate: Cooperative multi-agent path planning. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 87–94, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-1993-5. URL http://dl.acm.org/citation.cfm?id=2484920.2484938.

[24] Keith S Decker and Victor R Lesser. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems*, 1(02):319–346, 1992.

[25] Boris DeWilde, Adriaan W. Ter Mors, and Cees Witteveen. Push and Rotate: A complete Multi-agent Pathfinding algorithm. *Journal of Artificial Intelligence Research*, 51:443–492, 2014. ISSN 10769757. doi: 10.1613/jair.4447.

[26] K. Dresner and P. Stone. Multiagent traffic management: Opportunities for multiagent learning. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3898 LNAI:129–138, 2006. ISSN 03029743. doi: 10.1007/11691839_7.

[27] Edmund H Durfee. Scaling Up Agent Coordination Strategies Coordination Strategies. *Computer*, (July):39–46, 2001.

[28] Edmund H Durfee. Distributed problem solving and planning. In *ECCAI Advanced Course on Artificial Intelligence*, pages 118–149. Springer, 2001.

[29] Esra Erdem, Doga G. Kisa, Umut Oztok, and Peter Schüller. A general formal framework for pathfinding problems with multiple agents. *Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI 2013*, pages 290–296, 2013.

[30] EUROCONTROL. Airport collaborative decision-making (a-cdm) implementation manual, 2017.

[31] EUROCONTROL. European aviation in 2040, challenges of growth, annex 1, flight forecast to 2040, 2018.

[32] Aviation Intelligence Unit EUROCONTROL. Atfm regulation: a power for good, 2019.

[33] Ariel Felner, Meir Goldenberg, Guni Sharon, Roni Stern, Tal Beja, Nathan Sturtevant, Jonathan Schaeffer, and Robert C. Holte. Partial-expansion A* with selective node generation. *Proceedings of the 5th Annual Symposium on Combinatorial Search, SoCS 2012*, pages 180–181, 2012.

[34] Ariel Felner, Jiaoyang Li, Eli Boyarski, Hang Ma, Liron Cohen, T. K.Satish Kumar, and Sven Koenig. Adding heuristics to conflict-based search for multi-agent path finding. *Proceedings International Conference on Automated Planning and Scheduling, ICAPS*, 2018-June:83–87, 2018. ISSN 23340843.

[35] K Fines. Decentralized control for resilient airport surface movement operations, 2018.

[36] K Fines. Decentralized Control for Resilient Airport Surface Movement Operations. 2019.

[37] European Organization for the Safety of Air Navigation. Eurocontrol specification for a-smgcs services, 2018.

[38] Rajesh Ganesan, Poornima Balakrishna, and Lance Sherry. Improving quality of prediction in highly dynamic environments using approximate dynamic programming. *Quality and Reliability Engineering International*, 26(7):717–732, 2010. ISSN 07488017. doi: 10.1002/qre.1127.

[39] Chester Gong. Kinematic airport surface trajectory model development. *9th AIAA Aviation Technology, Integration and Operations (ATIO) Conference, Aircraft Noise and Emissions Reduction Symposium (ANERS)*, (September):1–11, 2009. doi: 10.2514/6.2009-7076.

[40] Schiphol Group. Noise. URL https://www.annualreportschiphol.com/our-results/people-environment-and-community/noise. Accessed: 22-01-2020.

[41] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:263–312, 2001. ISSN 10769757. doi: 10.1613/jair.855.

[42] Aerodromes ICAO and I Volume. Aerodrome design and operations, annex 14 to the convention on international civil aviation, 2018.

[43] International Civil Aviation Organization (ICAO). Annex 2: Rules of the air, 2005.

[44] Husni Idris, John-Paul Clarke, Rani Bhuva, and Laura Kang. Queuing Model for Taxi-Out Time Estimation. *Air Traffic Control Quarterly*, 10(1):1–22, 2002. ISSN 1064-3818. doi: 10.2514/atcq.10.1.1.

[45] Bunpei Irie and Sei Miyake. Capabilities of three-layered perceptrons. In *IEEE International Conference on Neural Networks*, volume 1, page 218, 1988.

[46] Jyh-Shing Jang. Anfis adaptive-network-based fuzzy inference system. *Systems, Man and Cybernetics, IEEE Transactions on*, 23:665 – 685, 06 1993. doi: 10.1109/21.256541.

[47] Yu Jiang, Zhihua Liao, and Honghai Zhang. A collaborative optimization model for ground taxi based on aircraft priority. *Mathematical Problems in Engineering*, 2013, 2013. ISSN 1024123X. doi: 10.1155/2013/854364.

[48] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[49] Mokhtar M. Khorshid, Robert C. Holte, and Nathan Sturtevant. A polynomial-time algorithm for non-optimal multi-agent pathfinding. *Proceedings of the 4th Annual Symposium on Combinatorial Search, SoCS 2011*, pages 76–83, 2011.

[50] Matthew S. Kistler and Gautam Gupta. Relationship between airport efficiency and surface traffic. *9th AIAA Aviation Technology, Integration and Operations (ATIO) Conference, Aircraft Noise and Emissions Reduction Symposium (ANERS)*, (September 2009), 2009. doi: 10.2514/6.2009-7078.

[51] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[52] Hanbong Lee and Hamsa Balakrishnan. Fast-time simulations of Detroit Airport operations for evaluating performance in the presence of uncertainties. *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, 2012. ISSN 21557195. doi: 10.1109/DASC.2012.6382349.

[53] Hanbong Lee, Waqar Malik, and Yoon C. Jung. Taxi-out time prediction for departures at charlotte airport using machine learning techniques. *16th AIAA Aviation Technology, Integration, and Operations Conference*, (June):1–11, 2016. doi: 10.2514/6.2016-3910.

[54] Ryan Luna and Kostas E. Bekris. Push and swap: Fast cooperative path-finding with completeness guarantees. *IJCAI International Joint Conference on Artificial Intelligence*, pages 294–300, 2011. ISSN 10450823. doi: 10.5591/978-1-57735-516-8/IJCAI11-059.

[55] Thomas W Malone, Kevin Crowston, et al. Toward an interdisciplinary theory of coordination. 1991.

[56] Robert Morris, Corina S Pasareanu, Kasper Luckow, Waqar Malik, Hang Ma, TK Satish Kumar, and Sven Koenig. Planning, scheduling and monitoring for airport surface operations. In *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[57] Luchtverkeersleiding Nederland. Aip netherlands: Aircraft parking / docking chart schiphol centre chart, . URL https://www.lvnl.nl/eaip/2020-01-16-AIRAC/html/index-en-GB.html. Accessed: 16-01-2020.

[58] Luchtverkeersleiding Nederland. Runway use, . URL https://en.lvnl.nl/environment/runway-use. Accessed: 22-01-2020.

[59] Luchtverkeersleiding Nederland. Weather conditions, . URL https://en.lvnl.nl/safety/achieving-safety/weather-conditions. Accessed: 22-01-2020.

[60] T.E.H. Noortman. *Agent-Based Modelling of an Airport's Ground Surface Movement Operation*. PhD thesis, 2018.

[61] International Civil Aviation Organization. Amofsg/10-sn no.14 - aerodrome meteorological observation and forecast study group (amofsg). URL https://www.icao.int/safety/meteorology/amofsg/AMOFSG%20Meeting%20Material/AMOFSG.10.SN.014.5.en.pdf. Accessed: 22-01-2020.

[62] John R Quinlan et al. Learning with continuous classes. In *5th Australian joint conference on artificial intelligence*, volume 92, pages 343–348. World Scientific, 1992.

[63] David B. Rappaport, Peter Yu, Katy Griffin, and Chris Daviau. Quantitative Analysis of Uncertainty in Airport Surface Operations. *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO)*, (September):1–16, 2009.

[64] S. Ravizza, J. A.D. Atkin, M. H. Maathuis, and E. K. Burke. A combined statistical approach and ground movement model for improving taxi time estimations at airports. *Journal of the Operational Research Society*, 64(9):1347–1360, 2013. ISSN 01605682. doi: 10.1057/jors.2012.123.

[65] Stefan Ravizza, Jun Chen, Jason A.D. Atkin, Paul Stewart, and Edmund K. Burke. Aircraft taxi time prediction: Comparisons and insights. *Applied Soft Computing Journal*, 14(PART C):397–406, 2014. ISSN 15684946. doi: 10.1016/j.asoc.2013.10.004. URL http://dx.doi.org/10.1016/j.asoc.2013.10.004.

[66] Paul Roling. Ae4445 airport operations: Runway design. Lecture slides, 2008.

[67] Malcolm Ryan. Exploiting subgraph structure in multi-robot path planning. *Journal of Artificial Intelligence Research*, 31:497–542, 2008. ISSN 10769757. doi: 10.1613/jair.2408.

[68] Malcolm Ryan. Constraint-based multi-robot path planning. pages 922–928, 2010.

[69] Qandeel Sajid, Ryan Luna, and Kostas E. Bekris. Multi-agent pathfinding with simultaneous execution of single-agent primitives. *Proceedings of the 5th Annual Symposium on Combinatorial Search, SoCS 2012*, pages 88–96, 2012.

[70] Schiphol. Traffic review 2018, 2018.

[71] Schiphol. Schiphol airport cdm operations manual, 2019.

[72] Guni Sharon, Roni Stern, Meir Goldenberg, and Ariel Felner. The Increasing Cost Tree Search for Optimal Multi-Agent Pathfinding. *Proceedings of the 4th Annual Symposium on Combinatorial Search, SoCS 2011*, 2(i):150–157, 2011.

[73] Guni Sharon, Roni Stern, Meir Goldenberg, and Ariel Felner. Pruning techniques for the increasing cost tree search for optimal multi-agent path finding. *Proceedings of the 4th Annual Symposium on Combinatorial Search, SoCS 2011*, pages 150–157, 2011.

[74] Guni Sharon, Roni Stern, Ariel Felner, and Nathan Sturtevant. Conflict-based search for optimal multi-agent path finding. *Proceedings of the 5th Annual Symposium on Combinatorial Search, SoCS 2012*, pages 97–104, 2012.

[75] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015. ISSN 00043702. doi: 10.1016/j.artint.2014.11.006. URL http://dx.doi.org/10.1016/j.artint.2014.11.006.

[76] Alexei Sharpanskykh. Lecture 1: Introduction to agents and multiagent systems. specification of multiagent systems. University Lecture, 2018.

[77] Alexei Sharpanskykh. Week 5: Learning and adaptation in multiagent systems. University Lecture, 2018.

[78] Alexei Sharpanskykh. Week 5: Multiagent planning and scheduling. University Lecture, 2019.

[79] Yoav Shoham and Moshe Tennenholtz. On the synthesis of useful social laws for artificial agent societies (preliminary report). pages 276–281, 01 1992.

[80] David Silver. Cooperative pathfinding. *AIIDE*, 1:117–122, 2005.

[81] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[82] Trevor Standley. Finding optimal solutions to cooperative pathfinding problems. *Proceedings of the National Conference on Artificial Intelligence*, 1:173–178, 2010.

[83] Trevor Standley and Richard Korf. Complete algorithms for cooperative pathfinding problems. *IJCAI International Joint Conference on Artificial Intelligence*, pages 668–673, 2011. ISSN 10450823. doi: 10.5591/978-1-57735-516-8/IJCAI11-118.

[84] Jan Renze Steenhuisen. *Coordinated Multi-Agent Planning and Scheduling*. Delft University of Technology, 2013. ISBN 9789461084408.

[85] Roni Stern, Nathan R Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, TK Satish Kumar, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Twelfth Annual Symposium on Combinatorial Search*, 2019.

[86] Pavel Surynek. A Novel Approach to Path Planning for Multiple Robots in Bi-connected Graphs. 2009.

[87] Pavel Surynek. On propositional encodings of cooperative path-finding. *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, 1:524–531, 2012. ISSN 10823409. doi: 10.1109/ICTAI.2012.77.

[88] Pavel Surynek. Makespan Optimal Solving of Cooperative Path-Finding via Reductions to Propositional. pages 1–40, 2016.

[89] Pavel Surynek, Ariel Felner, Roni Stern, and Eli Boyarski. Efficient SAT approach to multi-agent path finding under the sum of costs objective. *Frontiers in Artificial Intelligence and Applications*, 285(2012):810–818, 2016. ISSN 09226389. doi: 10.3233/978-1-61499-672-9-810.

[90] Pavel Surynek, Ariel Felner, Roni Stern, and Eli Boyarski. Modifying optimal SAT-based approach to multi-agent path-finding problem to suboptimal variants. *Proceedings of the 10th Annual Symposium on Combinatorial Search, SoCS 2017*, 2017-Janua:169–170, 2017.

[91] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[92] Karl Tuyls and Gerhard Weiss. Multiagent Learning: Basics, Challenges, and Prospects. pages 41–52, 2012.

[93] Heiko Udluft. *Decentralization in Air Transportation*. 2017. ISBN 9789055841745. doi: 10.4233/uuid.

[94] Iowa State University. [eham] amsterdam/schiph windrose plot. URL https://mesonet.agron.iastate.edu/sites/windrose.phtml?station=EHAK&network=NL__ASOS. Accessed: 22-01-2020.

[95] Manuela M Veloso, Joydeep Biswas, Brian Coltin, and Stephanie Rosenthal. Cobots: Robust symbiotic autonomous mobile service robots. In *IJCAI*, page 4423, 2015.

[96] Glenn Wagner. *Subdimensional Expansion: A Framework for Computationally Tractable Multirobot Path Planning*. 2015. doi: 10.1016/j.artint.2014.11.001.

[97] Glenn Wagner and Howie Choset. M*: A complete multirobot path planning algorithm with optimality bounds. *Lecture Notes in Electrical Engineering*, 57 LNEE:3260–3267, 2011. ISSN 18761100. doi: 10.1007/978-3-642-33971-4_10.

[98] Ko Hsin Cindy Wang and Adi Botea. Fast and memory-efficient multi-agent pathfinding. *ICAPS 2008 - Proceedings of the 18th International Conference on Automated Planning and Scheduling*, (Icaps):380–387, 2008.

[99] Ko Hsin Cindy Wang and Adi Botea. MAPP: A scalable multi-agent path planning algorithm with tractability and completeness guarantees. *Journal of Artificial Intelligence Research*, 42:55–90, 2011. ISSN 10769757. doi: 10.1613/jair.3370.

[100] L. . Wang. Fuzzy systems are universal approximators. In *[1992 Proceedings] IEEE International Conference on Fuzzy Systems*, pages 1163–1170, March 1992. doi: 10.1109/FUZZY.1992.258721.

[101] L. . Wang and J. M. Mendel. Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. *IEEE Transactions on Neural Networks*, 3(5):807–814, Sep. 1992. ISSN 1941-0093. doi: 10.1109/72.159070.

[102] Wenjie Wang and Wooi Boon Goh. Time optimized multi-agent path planning using guided iterative prioritized planning. *12th International Conference on Autonomous Agents and Multiagent Systems 2013, AAMAS 2013*, 2(3):1183–1184, 2013.

[103] Mathijs De Weerdt, Adriaan Mors, and Cees Witteveen. Multi-agent Planning An introduction to planning and coordination. pages 1–32, 2005.

[104] Jianan Yin, Yuxin Hu, Yuanyuan Ma, Yan Xu, Ke Han, and Dan Chen. Machine learning techniques for taxi-out time prediction with a macroscopic network topology. 09 2018. doi: 10.1109/DASC.2018.8569664.

[105] Jingjin Yu and Steven M. Lavalle. Planning optimal paths for multiple robots on graphs. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3612–3617, 2013. ISSN 10504729. doi: 10.1109/ICRA.2013.6631084.

[106] Jingjin Yu and Steven M. LaValle. Optimal Multirobot Path Planning on Graphs: Complete Algorithms and Effective Heuristics. *IEEE Transactions on Robotics*, 32(5):1163–1177, 2016. ISSN 15523098. doi: 10.1109/TRO.2016.2593448.