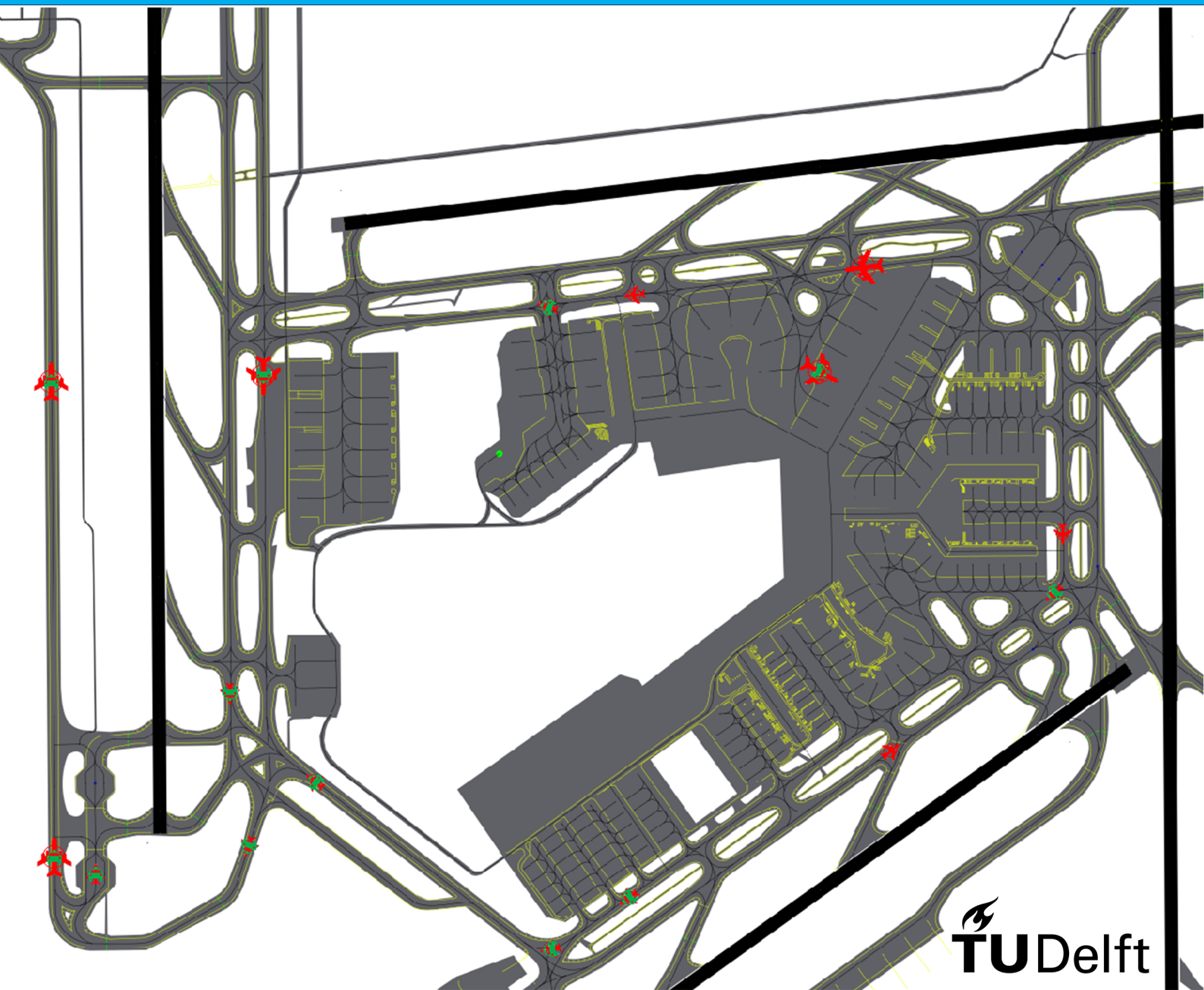


Design and Analysis of Tug-Enabled Engine-Off Taxiing Operation using Hierarchical Multi-Agent Planning

Master of Science Thesis

J.G. Kamphof



Design and Analysis of Tug-Enabled Engine-Off Taxiing Operation using Hierarchical Multi-Agent Planning

Master of Science Thesis

by

J.G. Kamphof

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday July 11th, 2022.

Student number: 4446615
Project duration: August 2021 – July 2022
Thesis committee: Dr.ir. B.F. Lopes Dos Santos
Dr. O.A. Sharpanskykh
Dr. B. Shyrokau
M.F. von der Burg

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Preface

Dear reader,

This thesis reports the work that has been carried out to obtain a Master of Science degree in Aerospace Engineering at Delft University of Technology. This work focuses on sustainable taxiing at airports in a novel concept of operations under hierarchical multi-agent planning. With the outcome of this project, I hope to provide more insights into the potential of both engine-off taxiing and multi-agent planning at airports.

In the process of the past months, I am very grateful for the supervision of dr. Alexei Sharpanskykh. His guidance, knowledge of the topic, feedback, and enthusiasm to discuss new concepts made this thesis project a valuable educational experience for me. At the same time, the freedom I got to shape my research was highly appreciated.

Furthermore, I would like to thank Malte von der Burg and Joost Soomers, with whom I collaborated very closely during the project. Joost's enthusiasm for the topic, hands-on approach, and drive for direct implementation was of great value. Malte's eye for detail, programming experience, and creativity contributed to the current state of simulation model and its future potential. In his co-supervisory role, Malte took all the time to provide me with critical feedback and to discuss endless ideas on new interesting projects that we could work on.

Aside from concluding my thesis, this work also marks the end of my educational journey at Delft University of Technology. I am grateful for the opportunities that I have had to develop myself, both within the curriculum and outside of it. Thanks to everyone who has been part of it!

Jorick Kamphof
Delft, June 2022

Contents

List of Figures	ix
List of Tables	xi
List of Abbreviations	xiii
I Scientific Paper	1
II Literature Study	
previously graded under AE4020	31
1 Introduction	33
2 Conventional Ground Surface Operations	35
2.1 General Overview of the Ground Surface Operation at an Airport	35
2.1.1 Key Components of the Ground Surface Operation	35
2.1.2 Airport Collaborative Decision Making (A-CDM).	36
2.1.3 Communications and Interactions.	39
2.2 Ground Surface Operations at Amsterdam Airport Schiphol	41
2.2.1 Runways	42
2.2.2 Taxiway network	43
2.2.3 Aprons and Gates	45
2.2.4 Remote holding locations	49
2.2.5 Service roads.	50
2.2.6 Concluding Remarks.	50
2.3 Traffic Management Techniques	51
2.3.1 Collaborative Pre-Departure Sequence Planning (CPDSP)	51
2.3.2 Gate holding	51
2.4 Concluding Remarks	52
3 Developments in Ground Surface Operations	53
3.1 Advanced Surface Movement Guidance and Control Systems (A-SMGCS)	53
3.1.1 Data required for the A-SMGCS system	54
3.1.2 Surveillance Service	54
3.1.3 Airport Safety Support Service	55
3.1.4 Routing Service	55
3.1.5 Guidance Service	55
3.2 Engine-Off Taxiing	55
3.2.1 Engine-Off Taxiing Techniques.	56
3.2.2 TaxiBot Concept and Technical Specifications	56
3.2.3 Main Differences TaxiBot Taxiing Operations and Conventional Taxi Operations	57
3.2.4 Challenges in the Implementation of TaxiBot	57
3.3 Novel Traffic Management Techniques	58
3.3.1 Outbound Holding.	58
3.3.2 Other strategies	59
3.4 Concluding Remarks	59
4 Modelling Ground Surface Operations	61
4.1 Complex Systems Modelling	62
4.1.1 Linear Programming and Search Algorithms.	62
4.1.2 Historical Data.	62
4.1.3 Markov Chains.	62

4.1.4	Agent-Based Modelling	63
4.2	Coordination in Multi-Agent Systems	63
4.3	Agent Based Modelling of Airport Ground Surface Operations at ATO.	64
4.3.1	Heiko Udluft: Decentralization in Air Transportation	65
4.3.2	T. Noortman: Agent-Based Modelling of an Airport's Ground Surface Movement Operation	65
4.3.3	K. Fines: Agent-Based Distributed Planning and Coordination for Resilient Airport Surface Movement Operations	67
4.3.4	B. Benda - Agent-Based Modelling and Analysis of Non-Autonomous Airport Ground Surface Operations.	67
4.3.5	Other ongoing studies at ATO	68
4.4	Concluding Remarks and Research Gap.	68
5	Multi-Agent Planning & Scheduling	69
5.1	Background on Path Finding	69
5.1.1	Shortest path finding with A*.	69
5.1.2	Multi-agent path finding (MAPF)	70
5.2	Classical approaches for MAPF	71
5.2.1	Extensions of A*	72
5.2.2	Conflict Based Search (CBS) and variants	73
5.2.3	Priority Based Solvers	75
5.3	MAPF for real-world robots	77
5.3.1	Perfect vs. imperfect plan execution	78
5.3.2	Uniform vs. non-uniform edge traversal times.	78
5.3.3	Point agents vs. geometric shaped agents	78
5.3.4	Agent Kinematics	80
5.3.5	Discretized time vs. Continuous time	81
5.4	Motion-Based MAPF Approaches	81
5.4.1	Single Agent Path Finding with Safe Interval Path Planning (SIPP)	82
5.4.2	Enhanced Conflict Based Search Continuous Time (ECBS-CT)	84
5.4.3	Continuous-Time Conflict Based Search (CCBS)	86
5.4.4	Continuous-Time Conflict Based Search with Kinematic Constraints (CCBS-kc)	87
5.5	Evaluation of Different Approaches	87
5.5.1	Evaluation Criteria	88
5.5.2	Trade-off.	88
5.6	Concluding Remarks	92
6	Research Proposal	93
6.1	Recap Literature Study	93
6.2	Research Objective and Research Questions	94
6.2.1	Prioritisation of research questions	95
6.3	Research Methodology	95
6.3.1	Methodological Steps	95
6.3.2	Research scope.	96
6.3.3	Planning	97
III	Supporting work	99
A	Model Elaboration	101
A.1	Layout Graph Generation Amsterdam Airport Schiphol.	101
A.1.1	Runway Placement at AAS	101
A.1.2	Construction of Abstracted Graph Layout	103
A.2	Route Planning	107
A.2.1	Activity Sequences	107
A.2.2	Priority Based Search	108
A.2.3	Safe Interval Path Planning.	110
A.2.4	Motion Generation.	112

A.3	Model Verification	114
A.3.1	Route Planning with Kinematics	114
A.3.2	Live Animation of Simulation Model.	115
A.3.3	Adherence to Layout Constraints	116
B	Simulation Setup	117
B.1	Algorithm Input Parameters	117
B.2	Vehicle Input Parameters	117
B.2.1	Kinematic Parameters	117
B.2.2	Engine Start Parameters	118
B.2.3	Shape Parameters	118
B.2.4	Wake Turbulence Separation (WTC) Parameters	118
B.2.5	Mapping to Simulated Aircraft Types.	119
B.3	Schedule Input Parameters	120
C	Simulation Results	123
C.1	Distributions Decoupling Standstill Times tt_{DC}	123
C.2	Heatmaps with Differences in Traversal Time per Edge	124
C.3	Heatmaps with Count of Edge Traversals per Day.	127
C.4	Analysis of Taxi Time Differences per Aircraft Type Group.	129
C.5	Average Taxi Time Variation for Shorter/Longer Decoupling	130
D	Statistical Elaboration on Experiments	133
D.1	Methods Used.	133
D.2	Comparisons under Variation of Engine Start Times	134
D.3	Comparisons under Variation of Decoupling Times.	136
E	Recommendations for Future Work	137
	Bibliography	139

List of Figures

1.1	Overview of the chapters in this literature study and the purpose(s) of each chapter. GSO refers to Ground Surface Operations.	34
2.1	Overview of the different stages in the ground surface operation, obtained from Jiang et al. [42].	36
2.2	Overview of the CDM Milestones [26].	38
2.3	Socio-technical system. Own work based on Fines [9] and Noortman [55].	39
2.4	Overview of the runways at AAS [68].	42
2.5	Percentage of total take-offs and landings, specified per runway. Own work based on "Baangebruikcijfers" [1].	43
2.6	Taxiways Alpha (green, clockwise), Bravo (orange, counterclockwise) and Quebec (red). Adapted from LVNL [49].	44
2.7	Overview of the existing piers at AAS [69].	45
2.8	Overview of the aircraft stand and signage used [66].	46
2.9	Satellite image of the west-side of the B-pier showing the ATC boundary by the red line. Adapted from Google Earth [20].	47
2.10	Standard push-back procedure at AAS based on Schiphol procedures [72].	48
2.11	Standard push-back paths from two example gates at AAS [72].	48
2.12	Holding positions at Amsterdam Airport Schiphol. Adapted based on map from OpenStreetMap contributors [57].	49
2.13	Overview of the Collaborative Pre-Departure Sequence Planning system at AAS [77].	51
3.1	Overview of the 4 services of the A-SMGCS system as specified by EUROCONTROL [24].	54
3.2	TaxiBot towing an aircraft at Schiphol Airport [45].	56
4.1	Centralized, Decentralized and Distributed Network [7].	61
4.2	Different forms of coordination in multi-agent systems. Obtained from [87].	64
4.3	Comparison of previous studies at ATO. Adapted from Soomers [90].	65
4.4	Effect of simplifying the airport layout [55].	66
4.5	Effect of replacing an apron by a single entry/exit node [55].	66
4.6	Effect merging a cluster of nodes to one node [55].	66
5.1	Types of conflicts. (a) edge conflict, (b) vertex conflict, (c) following conflict, (d) cycle conflict and (e) swapping conflict [94].	71
5.2	Example MAPF problem where two mice move from their start positions S1 and S2 to their goal positions G1 and G2 [84].	71
5.3	Constraint tree used in Conflict Based Search to resolve a conflict at D [84].	74
5.4	Situation where classic MAPF with point agents could result in a collision between the red aircraft and the green aircraft.	79
5.5	Example of successor generation in SIPP [58].	82
A.1	RMO North Runways. Green: departure runways, Red: arrival runways. Adapted based on LVNL [48]	102
A.2	RMO South Runways. Green: departure runways, Red: arrival runways. Adapted based on LVNL [48]	102
A.3	Part of the layout with: regular nodes (yellow), ramp nodes (red), ramp edges (red), taxiways (black), decoupling locations (blue), runways (grey).	103
A.4	Example push-back path from ramp E08 for two different aircraft types.	104
A.5	Original push-back path from ramp E08 for two different aircraft types. Obtained from [75] . . .	104
A.6	Decoupling locations per runway (RWY) shown as blue node.	105

A.7	Two blocked taxiways (red crosses) when Runway 36C is active for departures under RMO North or when Runway 18C is active for landings under RMO South. All aircraft to or from 36L/18R are directed over the taxiways south of Runway 36C/18C as shown in green. Northern direction is towards top of the image.	106
A.8	Example USI merging. Middle image under 2. based on image from Phillips and Likhachev [58].	110
A.9	Velocity profile for towed aircraft that pushes back, taxis, decouples and then departs from the airport.	114
A.10	Velocity heatmap for towed aircraft that pushes back, taxis, decouples and then departs.	114
A.11	Screenshot of the central taxiway area in the model of Amsterdam Airport Schiphol.	115
A.12	Screenshot of one of the gate areas (C/D-piers) in the model of Amsterdam Airport Schiphol.	115
A.13	Screenshot of one of the decoupling and runway areas (36C) in the model of Amsterdam Airport Schiphol.	116
A.14	Routes in test experiment when 36C is open and when 36C is closed. Note that scales are slightly different.	116
B.1	RMO North: Moving average of the spawn rate per 15 minutes for departure flights (top) and arrival flights (bottom).	120
B.2	RMO South: Moving average of the spawn rate per 15 minutes for departure flights (top) and arrival flights (bottom).	120
C.1	Decoupling standstill times under medium engine start duration.	123
C.2	RMO North - taxi time increase per edge	124
C.3	RMO North - taxi time decrease per edge.	124
C.4	RMO South - taxi time increase per edge	125
C.5	RMO South - taxi time decrease per edge	125
C.6	Decrease in traversal time for bay areas with many push-pull maneuvers.	126
C.7	Count of Traversals for Multi-Engine Taxiing and Tug-Enabled taxiing under RMO North.	127
C.8	Count of Traversals for Multi-Engine Taxiing and Tug-Enabled taxiing under RMO South.	128
C.9	Distribution of delta taxi times. Total outbound taxi time (top) and taxi time bay area (bottom) for different engine start durations.	129
C.10	Additional Taxi Time Differences for Departure Flights under variation of decoupling duration for RMO North.	130
C.11	Queue forming in scenario with longer decoupling duration in the big peak around 21:30h.	130
C.12	Additional Taxi Time Differences for Departure Flights under variation of decoupling duration for RMO South.	131

List of Tables

2.1	Two most relevant milestones of the A-CDM inbound phase	38
2.2	The milestones of the A-CDM turn around phase	38
2.3	The milestones of the A-CDM outbound phase	39
2.4	Preferred combinations of landing and take-off runways from 6:00 till 23:00 (a) and from 23:00-6:00(b) [10]. Preferred sequence 1 indicates the first preference, with the next preferences following.	43
2.5	Maximum turn around time before gate reallocation is considered at AAS based on Schiphol guidelines [71].	45
2.6	Three scenarios for the time before an aircraft can start taxiing after push-back and engine start.	49
2.7	Maximum wingspan restriction per position (N.A. indicating no restriction) [49].	50
5.1	Comparison of CBS algorithms based on the brc202d DAO map. Minimum score is 0, maximum score is 3.	89
5.2	Comparison of A* Based Prioritized Algorithms. Minimum score is 0, maximum score is 3.	90
5.3	Comparison of motion based approaches. Minimum score is 0, maximum score is 3.	91
6.1	Research Planning	97
B.1	Algorithm Parameters used for simulations.	117
B.2	Kinematic parameters per category. sType: simulated type, AC+TB: aircraft and tug combination, TMO: taxi mode of operation, MET: Multi-Engine Taxiing, TB: Tug-Enabled Taxiing, v_{max} : maximum velocity, acc : acceleration rate, dec : deceleration rate, r_{turn} : maximum turn radius, v_{turn} : maximum turn velocity, d_{TO} : minimum take-off distance.	118
B.3	Engine Start + Warm-Up times per sType category in seconds. sType: simulated type, ES: engine start.	118
B.4	Length, Width, radius offset (r_{offset}) and radius (r) per simulated ICAO type (sType).	118
B.5	Minimum Wake Turbulence Separation times in seconds. Based on RECAT-EU time-based separation minima for departure [62] and mapped to corresponding simulated aircraft types used in the simulation. Note that ICAO uses different names for the type categories than we do in our simulation.	119
B.6	Mapping from Aircraft Types to ICAO sType used in the simulations. Mapping based on ICAO reference codes [40].	119
B.7	Runway Schedule: Activation time of each (set of) runway combinations on both simulated days of operation.	121
B.8	Number of departure flights that use push-back, push-pull, and taxi-out. Split per simulated type group. RMO: Runway Mode of Operation.	121
C.1	Mean difference and median difference in total outbound taxi time tt_{dep} and taxi time on bay area tt_{BA} with respect to Multi-Engine Taxiing for Tug-Enabled Taxiing with a decoupling time of 2:00 minutes	129
D.1	Comparison of δtt_{dep} for shorter engine start times (-25%) and longer engine start times (+25%) with respect to a medium engine start duration for each decoupling duration. Results for runway 36L (a), 36C (b), 18L (c), and 24 (d). DC: decoupling, ES: engine start, RWY: Runway, S: small effect size, M: medium effect size. Significance level $0.05/2=0.025$	134
D.2	Comparison of δtt_{BA} for shorter engine start times (-25%) and longer engine start times (+25%) with respect to a medium engine start duration for each decoupling duration. Results for runway 36L (a), 36C (b), 18L (c), and 24 (d). DC: decoupling, ES: engine start, RWY: Runway, S: small effect size, M: medium effect size. Significance level $0.05/2=0.025$	134

D.3	Comparison of tt_{dep} for shorter engine start times (-25%) and longer engine start times (+25%) with respect to a medium engine start duration for each decoupling duration. Results for runway 36L (a) and 36C (b). DC: decoupling, ES: engine start, RWY: Runway. Significance level $0.05/2=0.025$	135
D.4	Comparison of tt_{dep} for shorter engine start times (-25%) and longer engine start times (+25%) with respect to a medium engine start duration for each decoupling duration. Results for runway 18L (a) and 24 (b). DC: decoupling, ES: engine start, RWY: Runway. Significance level $0.05/2=0.025$	135
D.5	Comparison of tt_{BA} for shorter engine start times (-25%) and longer engine start times (+25%) with respect to a medium engine start duration for each decoupling duration. Results for runway 36L (a), 36C (b), 18L (c) and 24 (d). Zero differences included in ranking. DC: decoupling, ES: engine start, RWY: Runway. Significance level $0.05/2=0.025$	135
D.6	Comparison of $tt_{dep} - EStime$ for shorter engine start times (-25%) and longer engine start times (+25%) with respect to a medium engine start duration for Multi-Engine Taxiing scenarios. Results for runway 36L (a), 36C (b), 18L (c), and 24 (d). ES: engine start, RWY: Runway, N: negligible effect size. Significance level $0.05/2=0.025$	135
D.7	Comparison of δtt_{dep} for shorter decoupling times (-25%) and longer decoupling times (+25%) with respect to a medium decoupling duration (2:00 min) for each engine start duration. Results for runway 36L (a), 36C (b), 18L (c), and 24 (d). DC: decoupling, ES: engine start, RWY: Runway, S: small effect size, M: medium effect size. Significance level $0.05/2=0.025$	136
D.8	Comparison of δtt_{BA} for shorter decoupling times (-25%) and longer decoupling times (+25%) with respect to a medium decoupling duration (2:00 min) for each engine start duration. Results for runway 36L (a), 36C (b), 18L (c), and 24 (d). Zero differences included in ranking. DC: decoupling, ES: engine start, RWY: Runway, N: negligible effect size. Significance level $0.05/2=0.025$	136

List of Abbreviations

A-CDM	Airport Collaborative Decision Making
A-SMGCS	Advanced Surface Movement Guidance and Control System
A-VDGS	Advanced Visual Docking Guidance System
AAS	Amsterdam Airport Schiphol
ABM	Agent Based Modelling
AEON	Advanced Engine Off Navigation
AIBT	Actual In Block Time
AIP	Aeronautical Information Package
ALDT	Acutual Landing Time
AOBT	Actual Off Block Time
ARDT	Actual Ready Time
ARST	Actual Start Up Request Time
ASAT	Actual Start Up Approval Time
ASP	Answer Set Programming
ASRT	Actual Start Up Request Time
ATC	Air Traffic Control
ATCO	Air Traffic Control Officer
ATM	Air Traffic Management
ATO	Air Transport and Operations
ATOT	Actual Take-Off Time
AXIT	Actual Taxiing In Time
BCBS	Bounded Conflict Based Search
CA*	Cooperative A*
CAT	Category
CATC	Conflicting ATC Clearances
CBS	Conflict Based Search
CBSw/p	Conflict Based Search with priority
CCBS	Continous Conflict Based Search
CCBS-kc	Continous Time Conflict Based Search with Kinematic Constraints
CISS	Central Information System Schiphol

CMAC	Conformance Monitoring Alerts for Controllers
CO-WHCA*	Conflict Oriented Windowed Hierarchical Cooperative A*
CO-WHCA*P	Conflict Oriented Windowed Hierarchical Cooperative A* with prioritization
CPDSP	Collaborative Pre-Departure Sequence Planning System
CT	Constraint Tree
CTOT	Calculated Take-Off Time
CWP	Controller Work Place
DAO	Dragon Age Origins
ECBS	Enhanced Conflict Based Search
ECBS-CT	Enhanced Conflict Based Search Continuous Time
EIBT	Estimated In Block Time
EPEA*	Enhanced Partial Expansion A*
ESUT	Engine Start Up Time
EXIT	Estimated Taxi-In Time
EXOT	Estimated Taxi-Out Time
FtG	Follow the Greens
GCBS	Greedy Conflict Based Search
GSO	Ground Surface Operations
HCA*	Hierarchical Cooperative A*
HMI	Human Machine Interface
ICAO	International Civil Aviation Organization
ICTS	Increasing Cost Tree Search
ID	Independence Detection
IFR	Instrument Flight Rules
ILP	Integer Linear Programming
kts	knots
LA-CBS	Large Agent CBS
LA-MAPF	Multi Agent Path Finding for Large Agents
LGA	New York La Guardia Airport
LVNL	Luchtverkeersleiding Nederland / Air Traffic Control the Netherlands
MA-CBS	Meta-Agent CBS
MAMP	Multi Agent Motion Planning
MAPF	Multi Agent Path Finding
MC-CBS	Multi Constraint CBS

MET	Multi-Engine Taxiing
MTTT	Minimum Turn Round Times
NB	Narrow Body
NMOC	Network Manager Operations Center
OD	Operator Decomposition
PBS	Priority Based Search
PT	Priority Tree
RASAS	Regulation Aircraft Stand Allocation Schiphol
RMCA	Runway Monitoring and Conflict Alerting
RWY	Runway
SAT	Boolean Satisfiability
SCIPP	Soft Conflict Safe Interval Path Planning
SESAR	Single European Sky Air Traffic Management Research
SID	Standard Instrument Departure
SIPP	Safe Interval Path Planning
SIPPwRT	Safe Interval Path Planning with Reservation Table
TB	Tug-Enabled Taxiing
TCL	Taxiway Center Lights
TOBT	Target Off-Block Time
TP-SIPPwRT	Token Passing Safe Interval Path Planning with Reservation Table
TSAT	Target Start Up Approval Time
TTOT	Target Take-Off Time
VDGS	Visual Docking Guidance System
VHF	Very High Frequency
WB	Wide Body
WHCA*	Windowed Hierarchical Cooperative A*
WTC	Wake Turbulence Category

I

Scientific Paper

Design and Analysis of Tug-Enabled Engine-Off Taxiing Operation using Hierarchical Multi-Agent Planning

J.G. Kamphof^{*}, M.F. von der Burg[†], O.A. Sharpanskykh[‡]

Delft University of Technology, Delft, The Netherlands

Abstract

Growth in air traffic demand and the increasing importance of reducing emissions cause airports to consider the improvement of their ground surface movement operations in terms of efficiency and environmental impact. Over the past years, engine-off taxiing techniques have been considered to reduce the environmental impact of airport taxiing operations. One promising technique is Tug-Enabled Engine-Off Taxiing, a concept in which aircraft are towed to the runway by an external towing truck. However, implementation of this concept changes operations at the airport and creates challenges to coordinate and control vehicles efficiently and safely. Although operational changes are mostly expected in the bay areas and around decoupling locations close to the runways, especially bay areas are often excluded or simplified in existing research. In this work, we therefore aim to analyze Tug-Enabled Engine-Off Taxiing by using a detailed airport layout that includes bay area movements, such as push-back and engine start. A novel hierarchical multi-agent planning approach is used to route aircraft over the airport as an alternative to the current Air Traffic Control System. Vehicle movements are planned by a routing algorithm that accounts for shapes and kinematics. Real-world flight schedules from two days, under different runway modes of operation, were used to simulate ground surface movement operations. We compared outbound Tug-Enabled Taxiing to Multi-Engine Taxiing (MET) operations regarding taxi time, taxi distance, and bay area taxi times. Aircraft engine start times and tug decoupling durations were varied between in short, medium, and long scenarios. Our results show that the difference in taxi time between Tug-Enabled Taxiing and MET largely depends on the engine start time, decoupling duration, and taxiing distance to the runway. For runways far away from the ramps, lower taxiing velocities and the need for tug decoupling resulted in longer outbound taxi times in the Tug-Enabled Taxiing scenarios. However, in scenarios with long engine start time combined with short decoupling duration, the increase in taxi time was largely balanced out by a reduction of bay area taxi time. This trend becomes even more visible for operations to runways close to the ramps, since the engine start time forms a larger share of the total MET taxi time. Inbound flights were not significantly influenced by the Tug-Enabled Taxiing concept for outbound aircraft.

1 Introduction

Large airports are facing challenges to improve the efficiency and to reduce the environmental impact of their surface movement operations. Despite the disruptions of COVID-19, EUROCONTROL forecasts air traffic to recover to pre-COVID levels and grow further in the coming years ([1], [2]). As infrastructural expansion of airports is not expected to be sufficient to accommodate this growth [2], increasing the efficiency of the surface movement operations has become a challenge. At the same time, airports are looking for ways to reduce their environmental impact. The aircraft taxi-in and taxi-out phase currently forms a significant share of the total airport emissions ([3], [4]). Engine-off taxiing techniques are therefore being considered to reduce the environmental impact of airport taxiing. The available techniques can be grouped in on-board systems, integrated in the aircraft, and external systems, that are only temporarily connected to the aircraft [5, 6]. Whereas on-board systems have the advantage that aircraft are independent from any external devices, these systems will add additional weight that reduces the efficiency of the in-flight phase, especially on long-haul flights. Besides that, the maximum taxiing velocity of existing onboard systems is lower than taxiing with the fastest available external systems [5]. For these reasons, we only focus on external systems in this study. One of the external systems is Tug-Enabled Taxiing, where an external towing truck tows the aircraft from the ramp to a location near the runways. TaxiBot is such a Tug-Enabled Taxiing system and is currently the only certified engine-off taxiing technique [7]. Preliminary operational tests with TaxiBots at Amsterdam Airport Schiphol (AAS) have shown fuel savings up to 68% in comparison to Multi-Engine Taxiing operations [8]. However, deployment of

^{*}MSc Student, Air Transport and Operations, Faculty of Aerospace Engineering, Delft University of Technology

[†]PhD Candidate, 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

Tug-Enabled Taxiing further increases the traffic volume and complexity at airports [6]. These concepts thus pose additional challenges for efficient and safe control of surface movement operations. Therefore, there is a need to analyze 1) the operational effects of the implementation of Tug-Enabled Taxiing in airport ground surface movement operations and, 2) novel approaches to increase the efficiency of surface movement operations.

Limited research has been done in the area of Tug-Enabled Taxiing at large airports. Main operational differences between the conventional Multi-Engine Taxiing operations and Tug-Enabled Taxiing operations exist. First, with Tug-Enabled Taxiing, aircraft are not required to start their engines at the bay area after push-back. The bay areas, that are usually blocked temporarily by aircraft starting their engines, can thus be vacated faster. Second, there is a need to decouple the tug from the aircraft after taxiing instead of after push-back. This introduces a new decoupling operation that needs to be performed near the runways. Tug-Enabled Taxiing is therefore expected to change the traffic flow both in the bay areas and around the decoupling locations. Benda [9] investigated the operational safety and efficiency consequences of Tug-Enabled Taxiing at AAS. The airport was represented as a grid-based environment and the bay area and ramps were excluded from the model. Push-back and engine start in the bay areas were therefore not modelled, although they considerably effect delays in planning systems [10]. Other studies in Engine-Off Taxiing consider groups of ramps [11] or a simplified pier model [12]. The inclusion of actual airport bay areas would allow a more realistic and extensive evaluation of Engine-Off Taxiing in general and Tug-Enabled Taxiing specifically, thereby allowing for a completer quantification of differences to Multi-Engine Taxiing. To study novel Engine-Off Taxiing techniques in detail, it is thus interesting to consider a detailed airport layout in which aircraft bay area movements are explicitly included.

Increasing the efficiency of surface movement operations in the conventional Air Traffic Control concept is challenging. Air Traffic Controllers use sub-optimal standard procedures such as standard taxiway directions to keep the workload within bounds. Alternatively, existing research therefore focuses on partly or fully automated planning and coordination techniques for airport ground surface movement operations. Whereas the classic Air Traffic Control (ATC) system is centralized around one controlling entity, several more decentralized concepts are being explored. Recent studies have shown that distributed control mechanisms form an effective and safe alternative control and coordination approach for airport movement operations [9, 13, 14]. However, these methods can, due to their distributed nature, optimize for local optima. Although fully centralized architectures usually have longer run times, they are more suitable to find a global optimum. Hierarchical distributed architectures combine both centralized and distributed approaches. A modular hierarchical distributed architecture for airport ground surface movement operations was therefore developed¹ and is shown in Figure 1. In this multi-layered approach, planning and conflict-avoidance is performed centrally. Local actors then instruct the plan to the moving vehicles and monitor the execution of the plan at a local level. Finally, instructions are executed by vehicles moving over the airport.

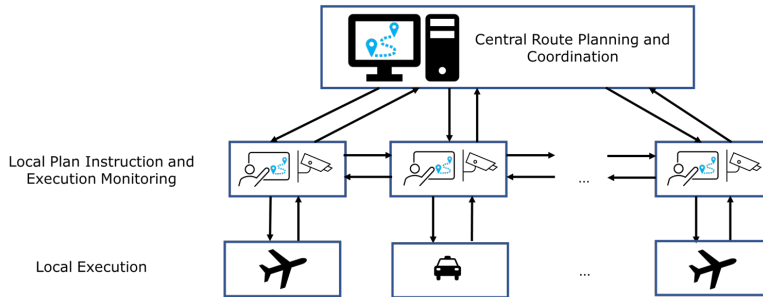


Figure 1: Hierarchical Distributed Architecture for Airport Surface Movement Operations Planning.

In this study, we develop a new concept for Tug-Enabled Taxiing by specifically focusing on the inclusion of bay areas in the airport layout and by including typical bay area operations such as push-back and engine start. Additionally, we further explore the concept of modular hierarchical multi-agent planning for airport ground surface movement operations. In order to review Tug-Enabled Taxiing, this concept will be compared to Multi-Engine Taxiing operations in the context of the hierarchical distributed architecture. The objective of this study is thus *to design and analyze a new concept of tug-enabled engine-off taxiing under hierarchical multi-agent planning, by using a detailed airport layout that includes bay area movements.*

To achieve this research goal, the following steps were followed. First, we developed a detailed airport layout that includes the bay areas and push-back paths at Amsterdam Airport Schiphol (AAS). Second, concepts of

¹Unpublished collaborative work with equal contribution by M.F. von der Burg (PhD candidate), J.G. Kamphof (MSc Student) and J.A.L. Soomers (MSc Student).

operations for Tug-Enabled Taxiing and Multi-Engine Taxiing were developed based on airport procedures and operational experts. Third, a Multi-Agent System (MAS) simulation model was developed in order to study the two concepts in the context of the hierarchical distributed control architecture. Fourth, an adapted version of Priority Based Search (PBS) and Safe Interval Path Planning (SIPP) was developed especially for the application on airport surface movement operations. Main modifications are the inclusion of aircraft kinematics and shapes. Fifth, verification and validation were performed. Finally, we analyzed 24 different scenarios to study the performance of Tug-Enabled Taxiing in comparison to Multi-Engine Taxiing based on taxi times, bay area taxi times and decoupling durations.

The structure of this paper is as follows. In section 2 the conceptual model and the case study with associated assumptions is presented. In section 3, we then describe the Multi-Agent System and Routing Algorithm in detail. Thereafter, verification and validation steps and outcomes are discussed in section 4. The experimental setup is then presented in section 5, with corresponding results in section 6. Finally, a discussion on the results and the methodology is provided in section 7, and conclusions are presented in section 8.

2 Conceptual Model and Case Study

This section presents the specification of the conceptual model for Multi-Engine Taxiing and Tug-Enabled outbound taxiing. First, the operational concepts and the main differences between the two concepts are described in section 2.1. Modelling assumptions and case study specifications for analyzing Tug-Enabled Taxiing at case study airport Amsterdam Airport Schiphol (AAS) will then be described in section 2.2.

2.1 Operational Concepts for Multi-Engine Taxiing and Tug-Enabled Taxiing

Multi-Engine Taxiing (MET): The concept of Multi-Engine Taxiing operations is based on existing procedures and is shown in Figure 2. The outbound operation is separated in three parts: 1) the push-back and (possibly) push-pull maneuver, 2) push-back truck decoupling and engine start, and 3) taxiing to the runway. We assume that both aircraft engines are required to be started before taxiing and that aircraft can commence the engine start procedure at the scheduled departure time from the ramp. Only in the case of a push-pull maneuver, engine start must be performed entirely after finishing the pull maneuver. For arrival flights, we assume that the taxi-in time from runway to the ramp is sufficient to perform engine cool down.

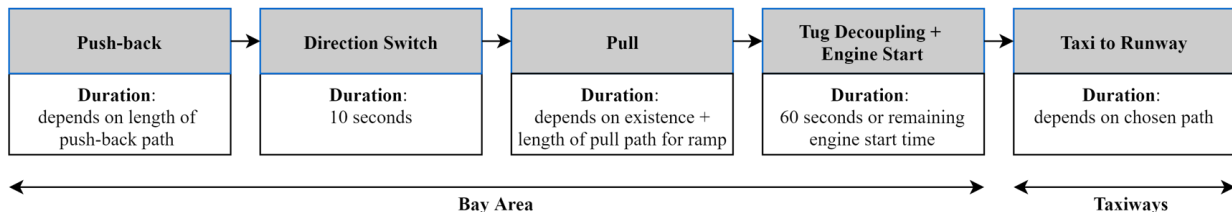


Figure 2: Concept of operations for outbound Multi-Engine Taxiing.

Tug-Enabled Taxiing (TB): The concept of Tug-Enabled Taxiing was based on operational expert interviews and the specifications of the reference TaxiBot system [7]. For the scope of this thesis, we consider Tug-Enabled Taxiing only for outbound flights. The reason is that more differences with respect to MET are expected in the bay areas for outbound flights using Tug-Enabled Taxiing than for inbound flights. An overview of this outbound operation is visualized in Figure 3. It is divided in four stages: 1) push-back and direction switch, 2) Tug-Enabled Taxiing to a decoupling location, 3) decoupling and all clear sign by the tug driver to the pilot, and 4) taxiing to the runway. The main differences with MET taxiing are that aircraft can directly start taxiing after push-back and that tug-decoupling needs to be performed at a dedicated decoupling location near the runway. It is assumed that engine start can be performed during Tug-Enabled Taxiing. If the duration of push-back, taxiing, and decoupling is shorter than the time needed to start the engines, aircraft wait till full engine start-up after decoupling. After decoupling, the tug driver gives an all-clear sign to the pilot to indicate that decoupling has finished. Then, it drives back to a tug base via either the service roads or the taxiways. For the scope of this thesis, we assume that tugs can perform this return movement independently and without interference with other traffic. Therefore, tugs that are not connected to an aircraft do not have to be accounted for by the routing algorithm. Furthermore, Tug-Enabled Taxiing operations are subject to different kinematic properties than MET operations. These will be described in the experimental setup in section 5.

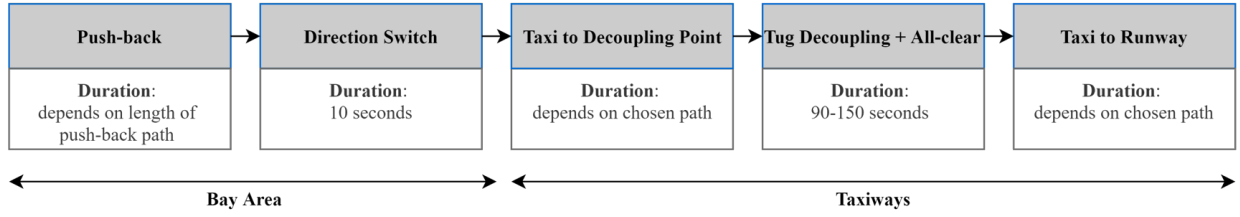


Figure 3: Concept of operations for outbound Tug-Enabled Taxiing

2.2 Tug-Enabled Taxiing at Amsterdam Airport Schiphol

For this study, we used outbound Tug-Enabled Taxiing at Amsterdam Airport Schiphol (AAS) as a case study. The reason is threefold. Firstly, AAS is one of Europe’s largest airports with a high degree of operational complexity and a high traffic load. Secondly, AAS has already performed first trials with the TaxiBot system, thereby offering first insights on the operational feasibility. Thirdly, an accurate description of standard push-back paths is available [15], thereby allowing for accurate modelling of the processes in the bay areas. In this section, we specify the conceptual model for AAS further, by looking at the Runway Mode of Operation (RMO) and decoupling locations. Finally, we end the section with final modelling assumptions.

The most commonly used runway modes of operation (RMO) at AAS are ‘RMO North’ and ‘RMO South’. When RMO North is active, flights depart from runways 36C and 36L and arrive at runway 06 and 36R. In comparison to that, flights depart from runway 24 and 18L and arrive at runway 18R and 18C when RMO South is active. The locations of these runways are shown in Appendix A as a reference. Outbound taxi times under RMO North are generally longer than under RMO South. From the perspective of reducing emissions, using outbound Tug-Enabled Taxiing under RMO North is therefore expected to lead to highest reductions. However, to evaluate the concepts under different conditions we analyze one day of operations on which RMO North is active and another day on which RMO South is active.

Decoupling locations are defined at AAS at predefined locations per runway. It is assumed that both the decoupling locations at the holding platforms and the decoupling locations at the taxiways can be used for decoupling. Exact locations of decoupling locations and the selection process of those locations are specified in Appendix A.

We furthermore formalize the following modelling assumptions:

- Each flight spawns with an origin and goal destination that is either a ramp, a set of runway entries, or a runway exit. Departing flights can choose an entry as long as their minimum take-off distance is respected. Arrival flights are assumed to always take the first feasible exit based on their landing distance.
- Path planning is performed with exact knowledge about the aircraft’s kinematics and all aircraft execute their instructions perfectly.
- Every aircraft that leaves the gate with a push-back or push-pull maneuver has a tug coupled to it at departure and there is an unlimited availability of tugs and push-back trucks.
- All flights use the standard push-back paths as defined in the operational procedures of AAS.
- The return movement of tugs does not have to be coordinated with other traffic, i.e. tugs disappear after decoupling.
- When using Tug-Enabled Taxiing, aircraft can start their engines during taxiing. If engine start time is not exceeded after decoupling, aircraft wait at the decoupling location for the remaining start time.
- All flights in the flight schedule are categorized as one of the 6 aircraft types from the ICAO aerodrome reference codes [16]. They are assumed to have a circular size. Radii and kinematics per ICAO type are specified in Appendix B.
- As a novel hierarchical multi-agent planning concept is applied, it is assumed that standard direction routing procedures that are currently in place at AAS do not have to be adhered to.

3 Multi-Agent System

In this section, the specification of the multi-agent system is described². A global overview of the hierarchical multi-agent system is shown in Figure 4. It includes three main building blocks: the environment, multiple agents, and the routing algorithm. The environment consists of an airport layout graph, a flight schedule database, and a routing plans database. The specification of the agents in the MAS is as follows: the Airport Operations Agent is at the top of the hierarchy and specifies the runway configuration and updates the flight schedule. Based on this information, the Routing Agent performs path planning by using the Routing Algorithm. The Localized Agents are then responsible for spawning the Moving Agents and instructing those according to the generated plans. Finally, the Moving Agents represent the vehicles that move over the airport by executing the instructions from the Localized Agents. As new aircraft continuously land and depart from the airport, we perform online path planning. The Routing Agent replans every h_{plng} minutes, for a time window of the next w_{plng} minutes. Furthermore, Localized Agents instruct the Moving Agent every h_{exec} minutes or when replanning is performed. In the remainder of the section, a detailed specification of the environment is provided in section 3.1. This is then followed by a description of the agent specifications in section 3.2. Finally, the algorithmic implementation of the routing algorithm is described in section 3.3.

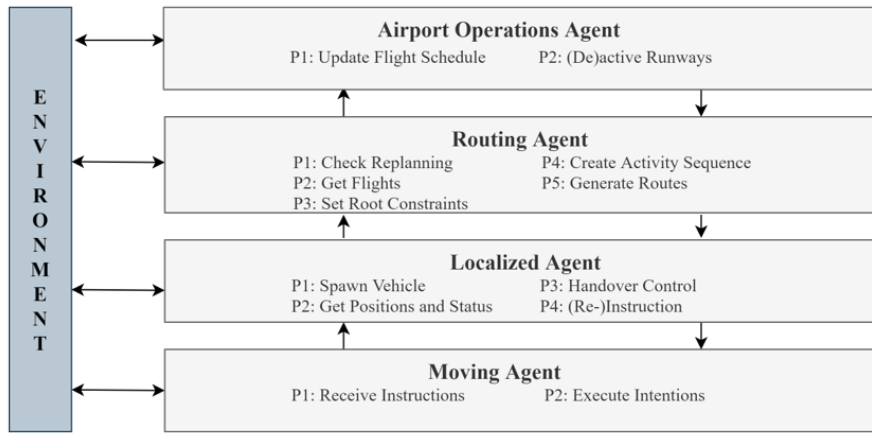


Figure 4: Overview of the Hierarchical Multi-Agent System with the properties (P) per agent. Routing Algorithm is part of the Routing Agent’s P5.

3.1 Environment Specification

The environment is composed of three objects: the airport layout, the flight schedule database, and the routing plans database. Each of those objects is explained in more detail below.

3.1.1 Airport Layout Graph

The airport taxiing infrastructure is represented by a graph $G = (V, E)$ with 1142 nodes (V) and 3188 directional edges (E), and is shown in Figure 5. Nodes represent taxiway intersections (yellow), decoupling locations (blue) or aircraft ramps (red). Each bidirectional taxiway segment between two nodes is constructed from two unidirectional edges that connect the nodes. Taxiway edges (black) are obtained from the actual locations of these taxiways at AAS. Ramp edges (red) represent the aircraft parking positions. To accurately model push-back procedures, push-back and/or push-pull edges are added for each of the 143 ramps in the layout. To ensure a realistic representation of the push-back and push-pull processes, these paths are accurately modelled based on standard push-back paths at Schiphol. Multiple push-back paths exist for some gates, as those are dependent on the size of the aircraft. The construction of push-back and push-pull paths is further elaborated in Appendix A. Finally, the graph contains decoupling points (blue) per runway that are used for tugs to decouple from the aircraft with Tug-Enabled Taxiing. The airport layout graph is fully accessible to the Airport Operations Agent and the Routing Agent. Furthermore, the airport layout graph is a static environmental object as it remains unchanged during the simulation.

²This section builds upon the collaborative work with equal contribution by M.F. von der Burg (PhD candidate), J.G. Kamphof (MSc Student) and J.A.L. Soomers (MSc Student).

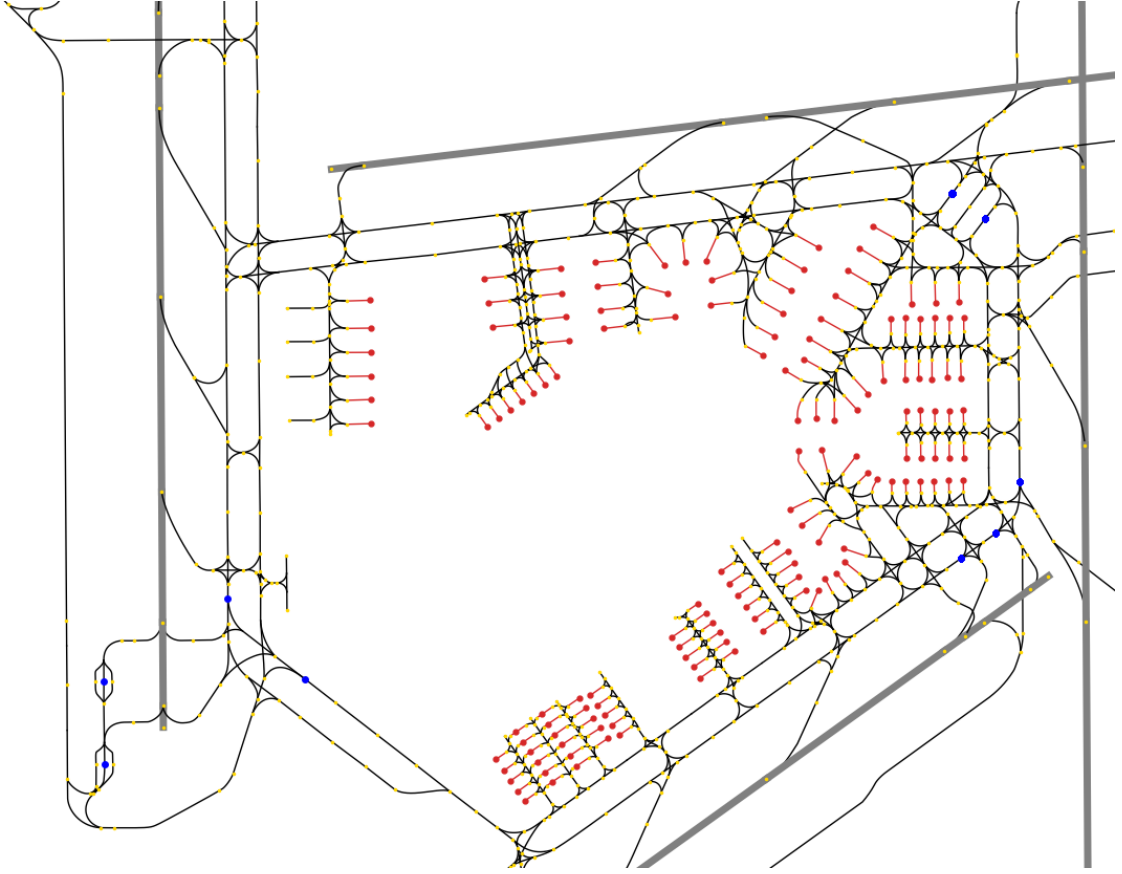


Figure 5: Part of the graph representation of Amsterdam Airport Schiphol. Taxiway edges (black), taxiway nodes (yellow), ramp nodes and edges (red), decoupling nodes (blue), and runways (grey). Full airport layout can be found in Appendix A.

3.1.2 Flight Schedule Database

The flight schedule has a vehicle ID, flight direction, ICAO type, runway (RWY) and ramp for each flight. Additionally, it stores the spawn time of each flight and the initial Taxi Mode of Operation (TMO) that can either be Tug-Enabled Taxiing (TB) or Multi-Engine Taxiing (MET). Table 1 shows a snapshot of the flight schedule. The flight schedule is stored as a dynamic environmental object, as the flight status can be changed by the Airport Operations Agent, the Routing Agent, and the Localized Agents.

Table 1: Flight schedule example. RWY: runway; sType: simulated ICAO type; TMO: taxi mode of operation; MET: Multi-Engine Taxiing; TB: Tug-Enabled Taxiing.

Vehicle ID	Spawn time	Direction	RWY	Ramp	sType	TMO	Status
AC-01	09:10	Arrival	06	E18	ICAO-C	MET	Finished
AC-02	09:18	Departure	36L	D94	ICAO-E	TB	Taxiing
AC-03	09:23	Arrival	06	D57	ICAO-C	MET	Routed
AC-04	10:23	Arrival	06	F08	ICAO-D	MET	Scheduled
AC-05	10:53	Departure	36L	G07	ICAO-C	TB	Planned

3.1.3 Routing Plans Database

The Routing Plans Database contains the most up to date version of the routes of all vehicles. Each plan is stored as a set of node, time, velocity (s, t, v) pairs for the remaining route. It is fully accessible to the Routing Agent, who updates the database once it has replanned the paths. The routing plans database is partially accessible to Localized Agents, who read out the route for the aircraft under their control.

3.2 Agent Specifications

3.2.1 Airport Operations Agent

The role of the Airport Operations Agent is to set the runway mode of operations with the corresponding constraints and to update the flight schedule. At least every rescheduling period h_{skdl} , the Airport Operations Agent updates the Flight Schedule for the next scheduling window w_{skdl} . The Runway Mode of Operations (RMO) is determined for the next 15 minutes. Some edges are not available under certain RMO configurations and are therefore constrained by this agent. The Airport Operations Agent has the following properties (P):

P1. Update Flight Schedule Property: For every time point t , the Airport Operations Agent checks if rescheduling is due based on h_{skdl} and the current time. If so, it updates the flight status from 'planned' to 'scheduled' for every flight within the flight schedule that spawns in the interval $[t, t + w_{skdl}]$.

P2. (De)Activate Runways Property: For every time point t , the Airport Operations Agent checks if there is an anticipated runway change in 15 minutes. If so, the Airport Operations Agent activates the anticipated runways and sets corresponding constraints. It does so by specifying a blockage time interval for the edges that need to be blocked for this RMO and passes those constraints to the Routing Agent. The edges that are blocked under each RMO are found in Appendix A.

3.2.2 Routing Agent

The role of the Routing Agent is to perform the central route planning. It generates a plan for all vehicles within the planning window w_{plng} at least every replanning period h_{plng} . For this, it uses the routing algorithm as specified in section 3.3. The Routing Agent obtains local information from the Localized Agents, performs central planning and then stores the new plan in the routing database in the environment. The properties of the Routing Agent are as follows:

P1. Check Replanning Property: At each timepoint t , the Routing Agent checks on an internal counter if replanning is due, or if it received any replanning requests from the Airport Operations Agent. If so, the internal counter is reset and the path planning is commenced by executing the Get Flights Property.

P2. Get Flights Property: This property involves interaction between the Routing Agent and the Environment. If replanning is due, the Routing Agent assesses all the flights with the status *scheduled* and *taxiing*. It adds the flights that are taxiing or that spawn within the planning window to the path planning. For each vehicle, the agent gets the ID, origin, initial velocity, destination, shape, and kinematic properties. It then executes the Set Root Constraints Property. Subsequently, the Routing Agent executes the Create Activity Sequence Property for those flights and updates the remaining activities for the flights that are already taxiing.

P3. Set Root Constraints Property: Involves interactions between the Routing Agent and the Airport Operations Agent. For every time point, if replanning takes place, the Routing Agent sets root constraints for the path planning. Root constraints include RMO constraints from the Airport Operations Agent and constraints that are inherited from the previous planning. The initial locations of all agents that are already taxiing are blocked in the root node.

P4. Create Activity Sequence Property: The Routing Agent defines a sequence of activities for each vehicle that is part of the planning. The path finding algorithm uses this sequence of activities as input. This is done so that activities can be forced to be on a vehicle's path in a predefined sequence. We describe the possible activities as follows:

- **Go-to activity:** Given a start location, goal location(s) and kinematics, the planning algorithm gets two degrees of freedom: time and route. This activity is used as input to taxi from one point at the airport to another point at the airport, for example for regular taxiing.
- **Follow activity:** With the follow activity the Routing Agent prescribes a predefined set of edges that must sequentially be part of the route. This means time is the only degree of freedom left in the planning algorithm, and that the path cannot be changed. This activity is used for push-back and push-pull maneuvers.

- **Wait activity:** The Routing Agent uses the wait activity to prescribe a waiting node location and waiting duration. It defines any particularities for the activity, such as the command to start the engine, to switch the direction of movement, or to decouple a tug from the aircraft.

When an activity sequence has been defined for each vehicle, the Routing Agent executes the Generate Routes Property.

P5. Generate Routes Property: After executing the previously described properties, the Routing Agent uses the Routing Algorithm to generate routes for all flights identified with the Get Flights Property. The algorithm is further described in section 3.3. When path planning is done, the Routing Agent stores the new routes for all vehicles in the routing database in the environment.

3.2.3 Localized Agents

The Localized Agents are responsible for instructing the Moving Agents and providing current positions of all taxiing vehicles to the Routing Agent. Whereas the Routing Agent plans the routes, the Localized Agents' role is to guide the Moving Agents along this route. They have a fixed location on the map and are always placed at a node. The Localized Agents have the following properties:

P1. Spawn Vehicle Property: For every time point, the Localized Agent checks if a new vehicle needs to be spawned. If that is the case, it copies the route from the routing plans database, spawns the agent, and hands it over to the next localized agent on the vehicle's route.

P2. Get Positions and Status Property: This property involves interactions between the Localized Agents and the Moving Agents. At each time point, the Localized Agents get the position, heading, and velocity of all vehicles under their control.

P3. Handover Control Property: This property involves interactions between 1. two Localized Agents or 2. Localized Agent and the Airport Operations Agent. At each time point, a Localized Agent checks whether a vehicle passed itself during the previous time step. If so, it hands over the control responsibility to the next, closest, Localized Agent on the vehicle's route. If a vehicle reaches its goal node, it is handed over to the Airport Operations Agent that removes the vehicle from the simulation.

P4. (Re-)Instruction Property: This property involves interactions between the Localized Agents and the Moving Agents. At each time point, the Localized Agent checks if the route of the vehicle under their control has changed with respect to the previous route for that vehicle. If the route changed, the localized agent triggers re-instruction. Additionally, re-instruction can be triggered by the internal counter of the Localized Agent which sends re-instructions every h_{exec} minutes. All instructions for the next w_{exec} minutes are communicated based on the distance from the current location where the instruction should be executed by the vehicle. Instructions can be of four types: velocity instructions, heading instructions, decoupling instructions, and engine start instructions.

3.2.4 Moving Agents

The role of the Moving Agents is to execute instructions that they received from the Localized Agents. In our research, we assume that the Moving Agents have the same kinematic properties as used by the Routing Agent in planning and that instructions are perfectly executed. We therefore assume no mismatch between planning and execution. The properties of the Moving Agent are the following:

P1. Receive Instructions Property: This property involves interaction between the Moving Agent and the Localized Agent controlling it. At every time point t , the Moving Agent checks if it has received any instructions from a Localized Agent. If so, it translates the instructions into intentions. These intentions are indexed by a distance from the current location from where they should be executed. The Moving Agent then executes the Execute Intentions Property.

P2. Execute Intentions Property: For every time step t , the Moving Agent checks its internal intentions list created in the Receive Instructions Property. It compares the distance on its trip meter to the distance from which the intentions should be executed. If it observes that the distance on the trip meter is equal to the start of the intended one, it executes the instruction.

3.3 Routing Algorithm

In this section, we present the adapted routing algorithm used by the Routing Agent. First, the requirements for the path planning algorithm and the available techniques are discussed in section 3.3.1. Second, the main principles of our applied routing algorithm for airport path planning are presented in section 3.3.2. Last, we elaborate on several main elements of the routing algorithm.

3.3.1 Requirements and Algorithm Selection

We formulate a set of requirements for the routing. Firstly, a conflict free route from ramp to runway or from runway to ramp shall be found for all vehicles. Secondly, the area of vehicles shall thereby be accounted for, as aircraft are large in comparison to the available space in the graph as specified in section 3.1. This introduces a new set of potential conflicts next to the classical center-center conflict. Hönig et al. [17] formalized those conflicts as vertex-vertex, vertex-edge, and edge-edge conflicts, where agents collide although their centers do not collide. Thirdly, vehicle kinematics should be considered in planning, as finite acceleration and deceleration play an important role for heavy vehicles such as aircraft. Finally, the algorithm should be able to handle the multiple sequential activities as defined by the routing agent.

Classical multi-agent path finding (MAPF) algorithms were used in previous studies to do path planning at airports [9, 14]. A limitation of these classical MAPF algorithms is that they assume vehicles to be a point that moves at constant velocity [18]. Those approaches were therefore not found to be suitable for our application. Instead, we further expand the Safe Interval Path Planning (SIPP) algorithm [19], that efficiently performs motion planning for single agents while avoiding dynamic obstacles. We adapt this algorithm to make it suitable for practical application in airport ground surface movement modelling with kinematics, shapes, and multiple activity planning. Furthermore, we use the principles of the Priority Based Search (PBS) [20] algorithm to efficiently plan for multiple agents.

3.3.2 Main Principles of the Routing Algorithm

This section outlines the main working principles of the Routing Algorithm used in this study. The high-level search, graph reservations, low-level search, and replanning concept will be elaborated below. Pseudocode of the algorithms is provided in Appendix A.

The Routing Algorithm consists of a two-level search, the high-level and the low-level. In the high-level search, we adopt the search from the Priority Based Search algorithm [20]. PBS maintains a priority tree and creates two new child nodes whenever it detects a conflict between two vehicles. In each child node, a new priority order is added, thereby constraining one of the two agents to avoid the other’s path. Based on sum of cost of the paths of all agents in each child node, the algorithm then chooses which priority order results in the lowest overall cost and expands this node next. In our implementation, we define cost as the sum of taxi times. As in PBS [20], we use a depth-first search in which the algorithm only backtracks when it cannot find a solution in the explored branch. The high-level search continues until the algorithm expands a child node without any collisions.

To respect the priority order set in the high-level search, we translate all paths into a set of graph reservations. In short, all aircraft temporarily block a set of edges during each movement between one node and another. The blockage times and set of blocked edges are dependent on the agent’s shape, velocity profile, and the shapes of other agents. The concept of graph reservations and conflict avoidance will be further discussed in section 3.3.3.

In the low-level search, we use an adapted version of Safe Interval Path Planning [19]. SIPP represents moving obstacles as collision intervals and subsequently defines a set of Safe Intervals (SIs) per graph location, representing time intervals during which an agent can occupy that location. Our implementation of SIPP searches for a single-agent path, thereby respecting the constraints from the high-level PBS search and the activity sequence as defined by the Routing Agent. We set the graph reservations in such a way, that the shape of the agent in the low-level search can be disregarded. Before performing the SIPP search, we translate the graph reservations into a set of Safe Intervals (SIs) and restrictions relevant for that vehicle. This translation is described in more detail in section 3.3.4. With those SIs defined, the algorithm efficiently computes a route with the time dimension included. Note that this is different from the original PBS algorithm, which uses space-time A* to perform low-level search. The details of the low-level search are described in section 3.3.5.

As described in the introduction of the MAS, we perform replanning every h_{plng} minutes for the next w_{plng} . In section 3.3.6 we therefore outline how replanning is handled in our Routing Algorithm.

3.3.3 Graph Reservations and Conflict Avoidance

To efficiently account for the shapes of agents, we translate each vehicles' plan into a set of reserved edges. The concept of graph reservations is based on the idea of translating the movement of a dynamic obstacle to a set of Unsafe Intervals (USIs) for a lower prioritized vehicle. Philips and Likhachev [19] denote these as collision intervals. We use three main inputs for making graph reservations: 1) the combined shape of one agent a_i and another agent a_j and 2) the velocity profile of agent a_i and 3) the distances between edges in the layout. This works as follows:

The combined shape represents the shape r_{a_i} of agent a_i , the shape r_{a_j} of agent a_j , and a predefined safety distance D . We simplify the shape of each agent to be a circle with radius $r = \max(W, L)$ where W is the vehicles' width and L the length. The combined shape r_s of two agents a_i and a_j is then calculated as follows:

$$r_s = r_{a_i} + r_{a_j} + D \quad (1)$$

Based on this r_s and the velocity profile of agent a_i , the start time and end time of the graph reservation are determined. We calculate four time points relative to each edge traversed by the center of agent a_i : t_1, t_2, t_3 and t_4 . Figure 6 shows these points for an example agent on edge AB. t_1 represents the time point that r_s touches the start node A of the edge A-B. t_2 is the moment r_s clears the start node of the edge. Finally, t_3 and t_4 represent the time points that the combined shape touches and clears the end point B of the edge A-B, respectively. If an aircraft stops at a node, we only define t_1 and t_2 , i.e., the moment r_s touches the node and the moment r_s clears the node. These time points are computed for every path entry of agent a_i and for every combination of radii of flights in the flight schedule.

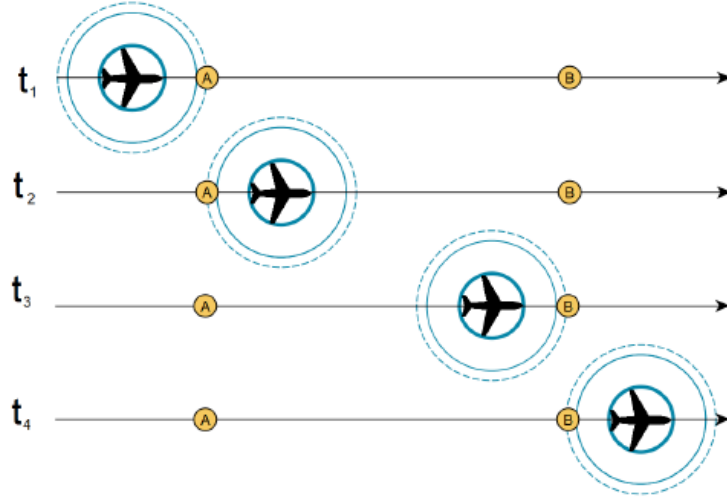
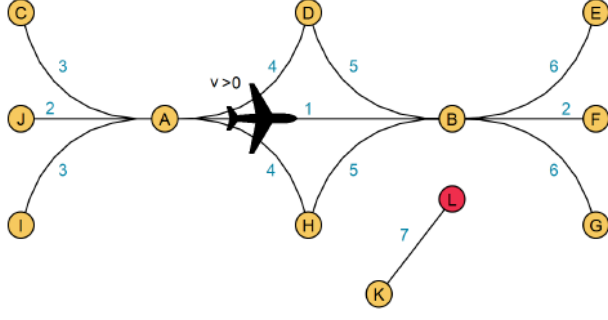
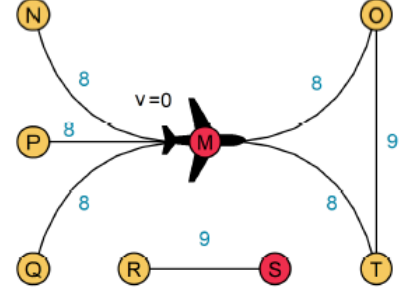


Figure 6: Calculated time points relative to each edge on an agent's route. t_1 : time point r_s touches the start of the edge, t_2 : time point r_s clears start of the edge, t_3 : time point r_s touches end of the edge, and t_4 : time point r_s clears end of the edge. Inner circle represents area of agent with radius r_{a_i} , middle circle has radius $r_{a_i} + r_{a_j}$ representing the area of both agents, and outer dashed circle represents the combined shape with radius r_s .

Once the t_1, t_2, t_3 and t_4 are determined for each edge and waiting node in the plan, the surrounding edges and nodes are considered. For each edge in the route of agent a_i , the algorithm determines the neighboring edges and the non-connected edges that are within a distance $(r_{a_i} + r_{a_j})$ from an edge. Furthermore, any nodes in the vicinity on which a waiting activity can be performed are reserved. We developed two reservation mechanisms to constrain the neighboring edges and nodes: 1) A reservation mechanism used when agent a_i has a non-zero velocity $v \neq 0$ and 2) a reservation mechanism used when agent a_i has a zero velocity due to an ongoing waiting activity. Furthermore, we distinguish between two types of graph reservations: Unsafe Intervals (USIs) and restrictions. USIs can prohibit other vehicles to enter the edge with a Do Not Enter (DNE) USI or to be present on the edge with a Do Not Persist (DNP) USI. Restrictions are additional constraints that must be respected by lower prioritized vehicles to prevent overtaking. The two reservation mechanisms are further explained below.



(a) Grouping of edges when $v = 0$



(b) Grouping of edges when $v \neq 0$

Figure 7: Edge reservations for zero and non-zero velocities.

1) Reservations with non-zero velocity $v \neq 0$

Figure 7(a) shows the reservation mechanism used to set constraints when agents are moving. Reservations are always set from edges on the vehicle's route one-by-one. In this example, we consider the movement from node A to node B in Figure 7 as an example. We first group the edges around edge A-B based on their locations. This grouping is shown in Figure 7, with the corresponding Unsafe Intervals (USIs) and restrictions per edge type defined in Table 2.

The edge labeled with a 1 always refers to the edge from which we are setting reservations. On this edge type 1, the constraining vehicle sets a DNE USI from $[t_1, t_2]$ in its own direction, prohibiting other vehicles to enter. In addition, it adds two restrictions for vehicles that use this edge to prevent overtaking: 1) if the other vehicle enters before t_1 , that vehicle should leave the edge before t_3 and 2) if the other vehicle enters after t_2 , that vehicle should not leave the edge before t_4 . The first restriction prevents the other vehicle to be overtaken by the own vehicle, whereas the second restriction prohibits the other vehicle to overtake the own vehicle. To prevent head-on collisions, the USI on the opposite edge of type 1 is from $[t_1, t_4]$. This USI restricts other vehicles to be present on that edge between t_1 and t_4 . The USIs on edge type 4 are the same: $[t_1, t_2]$ in the direction of traversal and $[t_1, t_4]$ in the opposite direction. Restrictions to prevent overtaking are also similar, but the t_3 and t_4 time points are calculated based on the distance on the type 4 edge. The USIs on edge types 5 and type 7 are from $[t_1, t_4]$ in both directions, restricting other vehicles to be present on those edges. Finally, all nodes on which a waiting activity can be performed and that are located within $(r_{a_i} + r_{a_j})$ meters of any point on the current edge, are constrained from t_1 to t_4 with a DNP USI.

Table 2: Edge types for reservation mechanism that is used when $v \neq 0$. $\Rightarrow$: in direction of traversal, $\Leftarrow$: in opposite direction, DNE: Do Not Enter, DNP: Do Not Persist. * For type 4 edges, t_3 and t_4 are calculated based on the traversed distance on the type 4 edge from the start node.

Edge Type	Description	Unsafe Intervals (USIs)	Restrictions
1	Edge currently traversed by agent a_i .	$\Rightarrow [t_1, t_2, DNE]$ $\Leftarrow [t_1, t_4, DNP]$	$\Rightarrow$ if entered before t_1 , leave before t_3 . $\Rightarrow$ if entered after t_2 , leave after t_4 .
2	Any other edge on the route of a_i within 150 meters from the current edge.	-	-
3	Edges to the start node of the current edge that make an angle $> 135^\circ$ with current edge.	-	-
4	Edges from the start node of the current edge that make an angle $< 135^\circ$ with current edge.	$\Rightarrow [t_1, t_2, DNE]$ $\Leftarrow [t_1, t_4, DNP]$	$\Rightarrow$ if entered before t_1 , leave before t_3^* . $\Rightarrow$ if entered after t_2 , leave after t_4^* .
5	Edges to the end node of the current edge that make an angle $< 135^\circ$ with current edge.	$\Rightarrow [t_1, t_4, DNP]$ $\Leftarrow [t_1, t_4, DNP]$	-
6	Type 4 edges of the next edge on agent a_i 's route.	-	-
7	Any non-connected edge to the current edge within $r_{a_i} + r_{a_j}$ meters of any point on the current edge.	$\Rightarrow [t_1, t_4, DNP]$ $\Leftarrow [t_1, t_4, DNP]$	-
Node L	Nodes on which a waiting activity can be performed within $r_{a_i} + r_{a_j}$ meters of any point on the current edge.	$[t_1, t_4, DNP]$	-

2) Reservations with zero velocity $v = 0$

Figure 7(b) shows the reservation mechanism used when velocities are equal to zero, and the USIs for this case are listed in Table 3. We set USIs on all outgoing edges from the moment the vehicle arrives at the node (t_1) until the moment that it clears the node (t_2). This prevents other vehicles to enter those edges during these intervals. In addition, edges within a distance $(r_{a_i} + r_{a_j})$ from the node are constrained from t_1 until t_2 in both directions for other vehicles to be present.

Table 3: Edge types for reservation mechanism that is used when $v = 0$. $\Rightarrow$: in direction to the node, $\Leftarrow$: in direction from the node, DNE: Do Not Enter, DNP: Do Not Persist.

Edge Type	Description	Unsafe Intervals (USIs)
8	Edges from the node where a_i is waiting.	$\Rightarrow [t_1, t_2, DNE]$
9	Any non-connected edge to the current edge within $r_{a_i} + r_{a_j}$ meters of any point on the current edge.	$\Rightarrow [t_1, t_2, DNP]$ $\Leftarrow [t_1, t_2, DNP]$
Node M	The node on which waiting is currently being performed by agent a_i .	$[t_1, t_2, DNP]$
Node S	Nodes on which a waiting activity can be performed within $r_{a_i} + r_{a_j}$ meters of any point on the current edge.	$[t_1, t_2, DNP]$

3.3.4 Conversion of Graph Reservations to Safe Intervals and Restrictions

Before starting the low-level search, we collect all relevant graph reservations for the agent a_i . Given the graph reservations and the Routing Agent constraints, a set of Safe Intervals, restrictions, and no-go edges is computed for a_i . First, the Unsafe Intervals and restrictions set for the ICAO type of the a_i are collected from all agents higher in priority. Constraints set by the routing agent are also translated in USIs and added to this set. Overlapping intervals are joined together and merged. If the time interval between the end of one USI and the start of another USI is smaller than the edge length l divided by the agent's maximum velocity V_{max} , we also merge the USIs into one, as the agent will never be able to traverse that edge. Finally, we create all intervals between $[t, t + w_{plng}]$ that do not overlap with the USIs, where t is the time of planning and w_{plng} is the planning window. The result is a set of Safe Intervals for this agent a_i and a set of restrictions that should be respected in the low-level search.

3.3.5 Low-Level Search

The low-level search is performed sequentially, based on the activity sequence generated by the Routing Agent. As described before, this sequence is created by the Routing Agent before planning and consists of go-to, follow, or wait activities. This section first presents the state definition and then elaborates on the activity-based search. In this search, we first generate a set of initial states for a single agent. From these initial states, we generate successors in two ways: regular successor generation and anticipatory successor generation. In regular successor generation we generate successors on the adjacent node of a vehicle's current node. In anticipatory successor generation, successors are generated on non-adjacent nodes that are within the agent's braking distance away from the current node.

State Definition: In general, a state is defined by a configuration and a Safe Interval: (cfg, SI) . We define the configuration as a set of the current node, the next node, velocity, and the activity ID $(N, N_n, v, actID)$. The safe interval SI corresponds to the edge (N, N_n) composed by the node and the next node. An exceptional case occurs for states with a velocity of 0 m/s. These are created with a Safe Interval on the node N instead of the edge. While not used for identification of the state, the cost, an arrival time, and a purpose (wait/move) are added to the state as additional information.

A. Initial State Generation: Before starting the activity-based search, the algorithm determines the start state for each agent. If a vehicle is taxiing, the initial state is generated at the node that this vehicle approaches, with the corresponding configuration and SI at that point. If the vehicle has an initial velocity of 0 m/s, we check two additional conditions: 1) are engines needed and started and 2) can this vehicle still wait at the current node. If the engines are needed but not started yet, the algorithm checks whether the start time of the engines is within the current SI on the node. If so, it generates a state. Else, the algorithm returns without a solution, as engine start is not possible and the vehicle is not able to perform any other motions without the engines started.

B. Regular Successor generation: Starting from the initial state, we generate successors based on the feasible motions that can be performed from this state. We assume that a vehicle always moves to a neighboring node as quickly as possible unless there is an USI or restriction that prevents the vehicle from doing so. Furthermore, we assume that the velocity profile on an edge is monotonic. In case there are no USI or restrictions on the neighboring edges of the next node, we optimize for minimum traversal time. Thereby we accelerate at the start of the edge and decelerate at the end of the edge. The result is a configuration and a SI. If the arrival time in that state is within a SI, we add the state as a successor state. In case there are USIs, we optimize for maximal final velocity on the edge and aim to arrive at exactly the start of the earliest Safe Interval. If there is no feasible motion, no successors are generated.

In the motion generation, we are bound to the vehicle's kinematic properties for the current activity and the velocity in the current state. A motion that is part of the follow-activity for push-back is for example constrained by a lower maximum velocity than regular taxiing in a go-to activity. In addition, vehicles that have maximum velocity in the current state, might not be able to decelerate enough to satisfy an USI on the next

edge or node. In this case, it might be required to start decelerating on the edge before the current state. To efficiently account for this, we anticipate based on the vehicle’s current velocity, braking distance, and USIs or restrictions within the braking distance.

C. Anticipatory Successor Generation: Whenever the algorithm opens a state, it explores the direct neighbors first. To anticipate on future USIs, restrictions, or turns for which agents need to slow down, anticipated successors are also generated. To do this, the algorithm considers all edges within the braking distance away from the node in the explored state. All successors that can be generated from that state are then considered. If no feasible motion is found from the explored state to the potential successor state, the algorithm checks whether the potential successor state can be reached from the state that it opened. If so, it generates the potential successor state at a non-neighbor node from the opened state.

3.3.6 Replanning

As discussed before, replanning is performed every h_{plng} minutes for the next w_{plng} minutes. As we plan with continuous time, agents can be in between two nodes when replanning is performed. Localized Agents estimate the arrival time of the agent on the next edge and pass this as input to the Routing Agent. The USIs and constraints set for this agent that already started are copied and used as root constraints to the next planning. In this way, other agents avoid the edges that this agent is blocking already.

4 Verification and Validation

Verification and validation of the simulation model was performed based on validation techniques and validation tests as described by Sargent [21]. To validate the conceptual model, operational experts from AAS were consulted as well as manufacturers from TaxiBots. During implementation, continuous verification was performed. The model was developed in different modules, allowing for the independent testing of the building blocks. In addition, assertion conditions were added to ensure correctness of the internal processes in the code and compiler errors were resolved. Visual animations were used to verify that the routes were executed as planned. Furthermore, small test scenarios were created to verify the model’s behavior in the bay areas for push-back, push-pull, and engine start, and for the decoupling locations to verify decoupling processes. Furthermore, the activity-based path planning was verified with small test scenarios ensuring correct timings and kinematics. With these scenarios, face validation was performed to ensure that the model performance was as expected. Finally, individual agent behavior was carefully followed throughout the system to ensure correctness.

5 Experimental Setup

This section presents the experimental setup that was used to compare Tug-Enabled Taxiing (TB) operations with Multi-Engine Taxiing (MET) operations in the hierarchical multi-agent concept. First, the hypotheses to be tested are stated and motivated in section 5.1. Second, the simulation plan is outlined in section 5.2. Third, key performance indicators are discussed in section 5.3. Finally, the proposed statistical evaluation is elaborated in section 5.4.

5.1 Hypotheses

The hypotheses are formulated at the global (G) and local (L) level, as stated below. We distinguish between departure flights and arrival flights. Furthermore, we formulate three sensitivity analysis (S) hypotheses to investigate how the results change under different engine start durations and decoupling times.

- G1** Taxi times for departure flights increase with TB in comparison to MET.
- G2** Taxi times for arrival flights decrease with TB in comparison to MET.
- L1** Bay area taxi times for outbound flights decrease with TB in comparison to MET.
- L2** Standstill time at decoupling locations increases for outbound flights with TB in comparison to MET.
- S1** The difference in outbound taxi time for TB with respect to MET decreases with an increase of engine start time.
- S2** The difference in outbound taxi time for TB with respect to MET increases with an increase of decoupling time.
- S3** Under variation of DC times, outbound taxi times in the TB scenarios differ more than only the increase in input.

The motivation for these hypotheses is as follows: the taxi times for departure flights are expected to increase (G1) because the maximum velocity of Tug-Enabled Taxiing is lower than with Multi-Engine Taxiing and because of expected congestion around the decoupling locations. For arrival flights, that do not use tugs in our experiments, we hypothesize a decrease in taxi time (G2) because these flights are expected to experience less hinder from departing aircraft that perform engine start during taxiing with Tug-Enabled Taxiing. We will test whether the bay area taxi time of departure flights indeed decreases with hypothesis L1.

A decrease in outbound taxi time difference (S1) is expected with an increasing engine start time because the benefits of starting the engine during taxiing will become more visible. For Tug-Enabled Taxiing scenarios in which decoupling takes longer we expect longer overall taxi times (S2) due to the need to stand still longer. Furthermore, we hypothesize that more congestion to forms up around decoupling locations with longer decoupling times, thereby increasing the outbound taxi time more than by only the difference in decoupling time (S3).

5.2 Simulation Plan

Scheduled data from two days of operations from 6:00-24:00 at Amsterdam Airport Schiphol (AAS) was used to compare Tug-Enabled Taxiing to Multi-Engine Taxiing. On the first day, RMO North is active, meaning that flights depart from runways 36L and 36C. On the other day, RMO South is active, in which flights depart from runways 24 and 18L. Like in actual operations, 2 departure runways are used in departure peaks and 2 arrival runways are used in arrival peaks. The number of arrivals and departures per day and per runway is shown in Table 4. The locations of the runways is shown Appendix A as a reference.

Table 4: Number of arrivals and departures per runway under RMO North (a) and RMO South (b).

(a) RMO North					(b) RMO South				
Runways	06	36R	36C	36L	Runways	18R	18C	24	18L
Count of Arrivals	436	220	0	0	Count of Arrivals	426	235	0	0
Count of Departures	0	0	245	430	Count of Departures	0	0	422	250

A ramp and runway are defined for each flight in the schedule. As discussed before, each flight is categorized as one of the six ICAO types. Kinematic parameters for both Multi Engine Taxiing (MET) and Tug-Enabled Taxiing (TB) are defined for all flights and are shown in Table 5.

Table 5: Kinematic Parameters for Multi-Engine Taxiing (MET) and Tug-Enabled Taxiing (TB).

Taxi Mode of Operation	Maximum Velocity	Max Turning Velocity	Acceleration rate	Deceleration rate
MET	15.0 $[m/s]$	5.0 $[m/s]$	0.25 $[m/s^2]$	0.75 $[m/s^2]$
TB taxiing	11.8 $[m/s]$	5.0 $[m/s]$	0.20 $[m/s^2]$	0.60 $[m/s^2]$

We perform analysis at different levels of detail so that both global performance and local performance are reflected in the results. First, all flights scheduled flights between 6:00-24:00 will be evaluated based on the Key Performance Indicators (KPIs) explained in section 5.3. We then perform an analysis that specifically focuses on the departure flights. Furthermore, we evaluate local KPIs in the bay areas and decoupling areas for departure flights, as those are expected to be influenced by the concept of Tug-Enabled Taxiing in particular.

We study twelve scenarios per simulated day of operations: three MET scenarios with short, medium, and long engine start times, and nine Tug-Enabled Taxiing scenarios with an additional variation of short, medium, and long decoupling times. We estimate the engine start takes 3 minutes for ICAO A,B,C types and 6 minutes for ICAO D,E,F types in the medium case. For short engine start times we use 75% of those values and for long engine start times 125%. Medium tug decoupling times for Tug-Enabled Taxiing are set to 2 minutes for all aircraft types and are also varied at 75% and 125% for short and long decoupling respectively. These variations are performed as these parameters are estimations. In real-life operations engine start times vary per individual aircraft based on factors such as engine type, airline procedures, and operational context. The decoupling times of 1:30 minutes and 2:00 minutes minutes are based on previous research and represent best case and worst case estimations from tug manufacturers [22], whereas the decoupling time of 2:30 minutes represents the average decoupling time in the Tug-Enabled Taxiing trials at AAS.

5.3 Key Performance Indicators

Tug-Enabled Taxiing was compared to Multi-Engine Taxiing by using a set of Key Performance Indicators (KPIs) on two levels. Global KPIs reflect the entire movement over the airport and local KPIs reflect only the movements in specific parts of the airport, such as the bay areas and the decoupling locations. While global KPIs are defined for both arrival flights and departure flights, local KPIs are only defined for departure flights, as these use Tug-Enabled taxiing. The set of KPIs is as follows:

- **Global:** Taxi time arrival flights (tt_{arr}) in minutes.
- **Global:** Taxi time departure flights (tt_{dep}) in minutes.
- **Global:** Taxi distance arrival flights (td_{arr}) in kilometers.
- **Global:** Taxi distance departure flights (td_{dep}) in kilometers.
- **Local:** Bay area taxi time for departure flights (tt_{BA}) in minutes.
- **Local:** Waiting time at decoupling location for departure flights (tt_{DC}) in minutes.

5.4 Statistical Evaluation

The data were not normally distributed, as the Shapiro-Wilk test [23] showed. This is as expected as the start point and end point for each vehicle are heterogeneous. Therefore, we used the non-parametric Wilcoxon signed-rank test with a significance level of $\alpha = 0.05$ to compare Tug-Enabled Taxiing scenarios to Multi-Engine Taxiing scenarios. For multiple comparisons, the significance level was corrected with the Bonferroni correction α/n to reduce the probability of a type I error, where n refers to the number of comparisons. To get a sense of the effect size and practical relevance of the results, the non-parametric Vargha-Delaney A-test [24] is performed if the Wilcoxon signed-rank test resulted in a significant difference. The A-test returns a value between 0.0 and 1.0, representing the probability that a randomly selected observation from one group is larger than a randomly selected observation from the other group. A value of 0.50 indicates that the medians are the same. Probabilities larger than 0.56, 0.64 and 0.71 represent a small, medium and large effect size, respectively. The same intervals are used for values smaller than 0.5.

6 Results

In this section, the results and their analysis are outlined in two parts. First, the results for the flights for the first input day under RMO North will be discussed in section 6.1. Second, the results for the second day on which RMO South was active will be discussed in section 6.2. To obtain the results, the conceptual agent-based model as discussed in section 3 was implemented in Python 3.9. All simulations were performed on a Windows 10 machine equipped with a 2.60 GHz Intel Core i7-9750H laptop with 16 GB RAM.

6.1 Evaluating RMO North

6.1.1 Global Trend Analysis

Tug-Enabled Taxiing operations were compared to the Multi-Engine Taxiing cases and statistical tests were performed. Table 6 shows the medians, first quartiles, and third quartiles of the Key Performance Indicators, as well as the significance of the difference of each KPI to the MET case. Light-red, orange and dark-red colors indicate a small, medium or large significant increase, respectively. Large significant decreases are marked in blue, and yellow colored cells indicate a significant result from the Wilcoxon signed-rank test with a negligible effect size. A negligible effect size therefore means that although the two groups are statistically different, the probability that a random observation from one group is larger than a random observation from the other group is negligible. Uncolored cells are non-significant differences. The values for Q1 and Q3 in Table 6 show us that the spread in data is relatively large. This makes sense, as the start and end points of the flights are not similar. Besides the global trends outlined in the following, we will therefore take a closer look at the results per departure runway in section 6.1.2

Table 6: Comparison of tug-enabled taxiing (TB) with Multi-Engine Taxiing (MET) with variation of Engine Start duration (ES) and Decoupling Duration. Significance level $0.05/3 = 0.0167$.

Short Engine Start: ICAO A, B, C: 2:15 minutes and ICAO D,E,F 4:30 minutes							
KPI	MET	TB1 - Decoupling 1:30 min		TB2 - Decoupling 2:00 min		TB3 - Decoupling 2:30 min	
	median (Q1, Q3)	median (Q1, Q3)	p-value, A-test	median (Q1, Q3)	p-value, A-test	median (Q1, Q3)	p-value, A-test
td_{arr}	1.91 (1.27, 2.55)	1.93 (1.26, 2.54)	0.63, (-)	1.93 (1.26, 2.54)	0.79, (-)	1.93 (1.26, 2.54)	0.60, (-)
td_{dep}	6.32 (3.15, 7.32)	6.34 (3.15, 7.43)	0.36, (-)	6.34 (3.25, 7.41)	0.10, (-)	6.34 (3.34, 7.43)	0.12, (-)
tt_{arr}	3.55 (2.53, 4.50)	3.47 (2.53, 4.46)	0.089, (-)	3.46 (2.50, 4.47)	0.068, (-)	3.46 (2.52, 4.46)	0.11, (-)
tt_{dep}	11.64 (8.62, 13.40)	13.61 (10.13, 15.66)	<0.001, (0.64, S)	14.29 (10.94, 16.41)	<0.001, (0.69, M)	14.98 (12.00, 17.16)	<0.001, (0.74, L)
tt_{BA}	3.22 (2.88, 4.32)	1.60 (1.32, 2.13)	<0.001, (0.056, L)	1.68 (1.33, 2.17)	<0.001, (0.071, L)	1.63 (1.33, 2.18)	<0.001, (0.085, L)
tt_{DC}	N/A	1.50 (1.50, 1.50)	N/A	2.00 (2.00, 2.00)	N/A	2.50 (2.50, 2.50)	N/A
Medium Engine Start: ICAO A, B, C: 3:00 minutes and ICAO D,E,F 6:00 minutes							
KPI	MET	TB1 - Decoupling 1:30 min		TB2 - Decoupling 2:00 min		TB3 - Decoupling 2:30 min	
	median (Q1, Q3)	median (Q1, Q3)	p-value, A-test	median (Q1, Q3)	p-value, A-test	median (Q1, Q3)	p-value, A-test
td_{arr}	1.91 (1.27, 2.54)	1.91 (1.26, 2.54)	0.83, (-)	1.93 (1.26, 2.54)	0.88, (-)	1.93 (1.26, 2.54)	0.91, (-)
td_{dep}	6.27 (3.15, 7.38)	6.34 (3.15, 7.43)	0.47, (-)	6.34 (3.23, 7.39)	0.33, (-)	6.34 (3.34, 7.43)	0.21, (-)
tt_{arr}	3.47 (2.53, 4.52)	3.46 (2.53, 4.46)	0.16, (-)	3.46 (2.50, 4.46)	0.099, (-)	3.46 (2.52, 4.46)	0.17, (-)
tt_{dep}	12.40 (9.55, 14.15)	13.59 (10.19, 15.70)	<0.001, (0.58, S)	14.29 (10.98, 16.42)	<0.001, (0.63, S)	14.97 (12.01, 17.17)	<0.001, (0.69, M)
tt_{BA}	3.97 (3.63, 5.13)	1.60 (1.32, 2.13)	<0.001, (0.024, L)	1.68 (1.33, 2.18)	<0.001, (0.033, L)	1.65 (1.33, 2.18)	<0.001, (0.049, L)
tt_{DC}	N/A	1.50 (1.50, 1.50)	N/A	2.00 (2.00, 2.00)	N/A	2.50 (2.50, 2.50)	N/A
Long Engine Start: ICAO A, B, C: 3:45 minutes and ICAO D,E,F 7:30 minutes							
KPI	MET	TB1 - Decoupling 1:30 min		TB2 - Decoupling 2:00 min		TB3 - Decoupling 2:30 min	
	median (Q1, Q3)	median (Q1, Q3)	p-value, A-test	median (Q1, Q3)	p-value, A-test	median (Q1, Q3)	p-value, A-test
td_{arr}	1.94 (1.27, 2.55)	1.91 (1.26, 2.54)	0.45, (-)	1.93 (1.26, 2.54)	0.54, (-)	1.93 (1.26, 2.54)	0.55, (-)
td_{dep}	6.27 (3.15, 7.42)	6.34 (3.15, 7.43)	0.54, (-)	6.34 (3.25, 7.39)	0.26, (-)	6.34 (3.34, 7.43)	0.17, (-)
tt_{arr}	3.51 (2.56, 4.62)	3.46 (2.53, 4.46)	0.031, (-)	3.46 (2.50, 4.47)	0.014, (0.48, N)	3.46 (2.52, 4.46)	0.028, (-)
tt_{dep}	13.19 (10.13, 15.05)	13.63 (10.17, 15.73)	0.056, (-)	14.32 (10.77, 16.38)	<0.001, (0.57, S)	14.91 (12.00, 17.13)	<0.001, (0.63, S)
tt_{BA}	4.72 (4.38, 5.90)	1.62 (1.32, 2.13)	<0.001, (0.015, L)	1.68 (1.33, 2.17)	<0.001, (0.020, L)	1.65 (1.33, 2.18)	<0.001, (0.034, L)
tt_{DC}	N/A	1.50 (1.50, 1.50)	N/A	2.00 (2.00, 2.00)	N/A	2.50 (2.50, 2.50)	N/A

MET: Multi Engine Taxiing, ES: Engine start duration, tt_{dep} : taxi time departures, td_{dep} : taxi distance departures, tt_{arr} : taxi time arrivals, td_{arr} : taxi distance arrivals, tt_{BA} : bay area taxi time, tt_{DC} : decoupling stand-still time. Effect sizes: N: Negligible, S: Small, M: Medium, L: Large. N/A: not applicable.

First, we consider the KPIs for total taxi times, split in arrival and departures. As shown in Table 6, 8 out of 9 Tug-Enabled Taxiing scenarios show an increase of departure taxi time (tt_{dep}) with respect to Multi-Engine Taxiing operations that is both statistically significant and relevant in size. No significant difference in tt_{dep} was found for the scenario with a long engine start duration and decoupling time of 1:30 minutes. This means that hypothesis G1 is rejected for this one scenario and supported for the other eight. The taxi distance (td_{dep}) for departure flights has not significantly changed in any of the nine Tug-Enabled Taxiing scenarios. This means that the increase in taxi time is mainly due to aircraft taking longer over the same taxi distance. We study the reason for this behavior further in the end of this subsection.

It can be observed in Table 6 that the effect sizes for the change of tt_{dep} are smaller for cases where engine start takes long than for cases where engine start time is short. For the Tug-Enabled Taxiing scenarios with a decoupling time of 2:00 minutes (TB2), we for example observe a shift from a medium increase in tt_{dep} of 2:39 minutes with short engine start, to a small increase of 1:08 minutes with long engine start. This suggests that the differences in tt_{dep} between Tug-Enabled Taxiing and Multi-Engine Taxiing are smaller when a longer engine start duration is required. This result is expected, as the benefit of starting the engines during taxiing makes the Tug-Enabled Taxiing concept more favorable when engine start is long. These results therefore support hypothesis S1. At the same time, global trends show that the effect size of the difference in tt_{dep} generally increases with larger decoupling times, thereby supporting our hypothesis S2. With a short engine start, for example, we observe a small increase of 1:58 minutes for a decoupling time of 1:30 minutes (TB1) to a large increase in tt_{dep} of 3:20 minutes for a decoupling time of 2:30 minutes (TB3). Also this effect is as expected, as an increasing decoupling duration means that the aircraft-tug combinations have to stand still longer in those cases. The increase in median with more than only the difference in decoupling times between those scenarios, indicates the presence of propagating effects of the increased decoupling time in the system. This will be further investigated runway level in section 6.1.2. For taxi time of arrivals flights, tt_{arr} , we do not observe a significant difference of relevant size in any of the Tug-Enabled Taxiing scenarios. This suggests that those flights are not influenced by the fact that departure flights use tugs instead of MET. Our hypothesis G2, that tt_{arr} decreases in the TB scenarios, is thus rejected for RMO North.

As a second step, we consider the bay area taxi times (tt_{BA}). From Table 6 we observe that tt_{BA} reduces with a large effect size in all Tug-Enabled Taxiing scenarios with respect to their MET base cases. Indeed, Table 6 shows that the median tt_{BA} reduces with 1:32 minutes in the scenario with short engine start and decoupling duration of 2:00 minutes (TB2). In the scenario with medium engine start duration, this effect is larger with a reduction of tt_{BA} of 2:17 minutes. The reduction is the largest with a long engine start duration, where the difference in median tt_{BA} is 3:02 minutes. Comparison of median tt_{BA} for the Tug-Enabled Taxiing

scenarios with the same decoupling duration does not show a difference of considerable size under variation of engine start time. However, aircraft using MET have to stand still longer before taxiing when engine start takes longer, thereby increasing the bay area taxi time. This is clearly reflected by the increase in tt_{BA} that can be observed in Table 6 within the three MET scenarios. We can thus assign the increasing effect sizes of tt_{BA} to the fact that the Multi-Engine Taxiing scenarios get worse. Therefore, the relative advantages of Tug-Enabled Taxiing over MET are reflected more clearly for large engine start durations.

Furthermore, we observe in Table 6 that the medians, Q1, and Q2 for decoupling standstill times (tt_{DC}) are equal to the specified decoupling time in each of the nine Tug-Enabled Taxiing scenarios. This suggests that, in general, aircraft have enough time to start their engines while taxiing to the decoupling location and performing decoupling. If this would not be generally the case, the medians would be higher, reflecting aircraft wait after decoupling. The fact that departure taxi times are generally longer than the engine start time explains this result. We provide more insights in the distribution of tt_{DC} under different runway configurations in Appendix C.

A summary of the global trends for RMO North is visualized in Figure 8. This figure shows how the total outbound taxi time (tt_{dep}) is distributed over the bay area, the taxiways, and standstill at the decoupling location in the different scenarios. Looking at the Tug-Enabled Taxiing scenarios with respect to Multi-Engine Taxiing in Figure 8, we observe that the share of each taxiing phase with respect to the total taxi time changed. Whereas the bay area taxi time, shown in purple, reduced for Tug-Enabled Taxiing in all cases, the other taxi time (green) increased for all Tug-Enabled Taxiing scenarios. The decoupling duration is added as an additional element in all TB scenarios, as is seen in the yellow bars. The reason for the decrease in bay area taxi time was previously discussed. The increase of other taxi time in the TB scenarios with respect to the MET scenarios is explained by the lower maximum taxiing velocity of 11.8 m/s for Tug-Enabled Taxiing with respect to 15 m/s in the MET scenarios. Heatmaps with the average increase and decrease per edge are reported in Appendix C and further support this observation. Besides that, Figure 8 shows a slight increase in the average other taxi time within the TB scenarios with the same engine start duration. This suggests that the tt_{dep} increases with more than only the difference in the specified decoupling time. It will be tested in the next section if this is indeed the case.

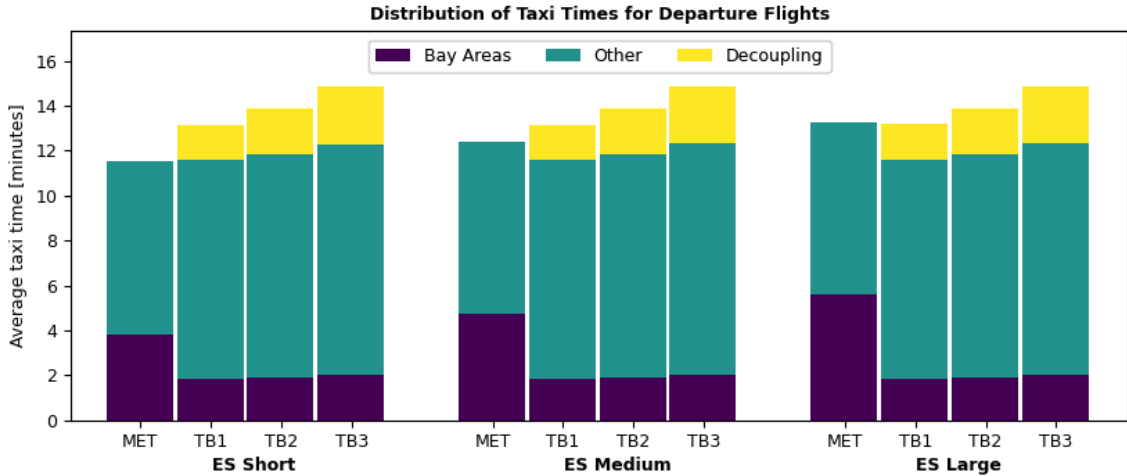


Figure 8: Distribution of outbound taxi time for RMO North. MET: Multi-Engine Taxiing, TB: Tug-Enabled Taxiing. TB1: Decoupling time of 1:30 min, TB2: Decoupling time of 2:00 min, TB3: Decoupling time of 2:30 min.

6.1.2 Runway Specific Analysis: 36L and 36C

The results for departure runways 36L and 36C, that are active under RMO North, are shown in Table 7 and Table 8. For flights to both runways 36L and 36C, we observe the same patterns as in the global data regarding the observations for tt_{dep} and tt_{BA} . Again, tt_{dep} is found to increase in 8 out of 9 Tug-Enabled Taxiing scenarios. For longer engine start times, we again see that the effect sizes of the increase in tt_{dep} become smaller, whereas these become larger for longer engine start durations. On a local level, we observe the bay area taxi time tt_{BA} reduces in all nine Tug-Enabled Taxiing scenarios with similar magnitude as in the global trend. In contrast to the global trend, we observe a small increase in taxi distance td_{dep} in 5 out of 9 Tug-Enabled Taxiing scenarios for runway 36C. However, the difference in medians with respect to MET is smaller than 250 meters in all cases, which is small with respect to distances at airports. We indeed see minor changes in the routes chosen around

the decoupling locations in the heat maps reported in Appendix C.

Table 7: Comparison of tug-enabled taxiing (TB) and Multi-Engine Taxiing (MET) for departure flights from RWY 36L. Significance level $0.05/3 = 0.0167$.

KPI		MET	TB1 - Decoupling 1:30 min		TB2 - Decoupling 2:00 min		TB3 - Decoupling 2:30 min	
		median	median	p-value, A-test	median	p-value, A-test	median	p-value, A-test
ES Short	td_{dep} [km]	6.98	7.03	0.84, (-)	7.02	0.93, (-)	7.02	0.42, (-)
	tt_{dep} [min]	12.82	14.94	<0.001, (0.73, L)	15.56	<0.001, (0.78, L)	16.18	<0.001, (0.82, L)
	tt_{BA} [min]	3.20	1.66	<0.001, (0.056, L)	1.68	<0.001, (0.068, L)	1.63	<0.001, (0.073, L)
ES Medium	td_{dep} [km]	6.97	7.03	0.74, (-)	7.02	0.87, (-)	7.02	0.51, (-)
	tt_{dep} [min]	13.51	14.95	<0.001, (0.62, S)	15.56	<0.001, (0.70, M)	16.13	<0.001, (0.75, L)
	tt_{BA} [min]	3.96	1.63	<0.001, (0.019, L)	1.68	<0.001, (0.032, L)	1.64	<0.001, (0.038, L)
ES Long	td_{dep} [km]	6.98	7.03	0.99, (-)	7.02	0.68, (-)	7.03	0.42, (-)
	tt_{dep} [min]	14.24	14.97	0.025, (-)	15.54	<0.001, (0.62, S)	16.13	<0.001, (0.68, M)
	tt_{BA} [min]	4.68	1.63	<0.001, (0.011, L)	1.69	<0.001, (0.021, L)	1.64	<0.001, (0.026, L)

MET: Multi-Engine Taxiing, TB: Tug-Enabled Taxiing, ES: Engine start duration, td_{dep} : taxi distance departures, tt_{dep} : taxi time departures, tt_{BA} : bay area taxi time. ES Short: 2:15 (ICAO A-C) and 4:30 minutes (ICAO D-F), ES Medium: 3:00 and 6:00 minutes, ES Long: 3:45 and 7:00 minutes.

Table 8: Comparison of tug-enabled taxiing (TB) and Multi-Engine Taxiing (MET) for departure flights from RWY 36C. Significance level $0.05/3 = 0.0167$.

KPI		MET	TB1 - Decoupling 1:30 min		TB2 - Decoupling 2:00 min		TB3 - Decoupling 2:30 min	
		median	median	p-value, A-test	median	p-value, A-test	median	p-value, A-test
ES Short	td_{dep} [km]	2.62	2.68	0.006, (0.53, N)	2.73	<0.001, (0.561, S)	2.76	<0.001, (0.58, S)
	tt_{dep} [min]	7.69	9.27	<0.001, (0.71, M)	9.93	<0.001, (0.78, L)	10.71	<0.001, (0.85, L)
	tt_{BA} [min]	3.22	1.58	<0.001, (0.055, L)	1.68	<0.001, (0.076, L)	1.65	<0.001, (0.11, L)
ES Medium	td_{dep} [km]	2.62	2.68	0.036, (-)	2.72	0.003, (0.56, S)	2.75	<0.001, (0.58, S)
	tt_{dep} [min]	8.58	9.27	<0.001, (0.59, S)	9.93	<0.001, (0.68, M)	10.76	<0.001, (0.78, L)
	tt_{BA} [min]	3.97	1.58	<0.001, (0.033, L)	1.68	<0.001, (0.034, L)	1.65	<0.001, (0.068, L)
ES Long	td_{dep} [km]	2.62	2.68	0.014, (0.53, N)	2.73	0.002, (0.55, N)	2.78	<0.001, (0.58, S)
	tt_{dep} [min]	9.24	9.29	0.083, (-)	9.90	<0.001, (0.58, S)	10.75	<0.001, (0.70, M)
	tt_{BA} [min]	4.72	1.60	<0.001, (0.022, L)	1.68	<0.001, (0.017, L)	1.65	<0.001, (0.050, L)

MET: Multi-Engine Taxiing, TB: Tug-Enabled Taxiing, ES: Engine start duration, td_{dep} : taxi distance departures, tt_{dep} : taxi time departures, tt_{BA} : bay area taxi time. ES Short: 2:15 (ICAO A-C) and 4:30 minutes (ICAO D-F), ES Medium: 3:00 and 6:00 minutes, ES Long: 3:45 and 7:00 minutes.

In order to test how the variation of engine start time and decoupling time influences the Tug-Enabled Taxiing scenarios under RMO North, we further investigated the differences between the nine TB scenarios. As shown in Table 7 and Table 8, the medians for tt_{BA} are similar for all 9 Tug-Enabled Taxiing scenarios, thereby suggesting that neither engine start nor decoupling time variation influences the time spend on the bay area. This is confirmed by results from the Wilcoxon signed-rank test with a significance of $p < 0.001$, as reported in Appendix D. Both engine start time and decoupling duration do thus not significantly influence tt_{BA} in the Tug-Enabled Taxiing scenarios under RMO North. Furthermore, we do not observe a considerable difference in median tt_{dep} when engine start time increases, which was also confirmed by the Wilcoxon Signed-Rank test. This means that engine start does also not significantly change the tt_{dep} of Tug-Enabled Taxiing. However, all rows of Table 7 and Table 8 show that the median of tt_{dep} increases for longer decoupling durations. This was confirmed by the Wilcoxon signed-rank with a significance of $p < 0.001$. The result is as expected, as aircraft are forced to stand still longer in those scenarios. The fact that the median increases by more than the difference in decoupling duration, shows us that there are other factors besides the longer decoupling duration. Table 9 shows the results of a comparison of shorter decoupling and longer decoupling duration with respect to a medium decoupling time of 2:00 minutes. For 5 scenarios in Table 9 we observe a small difference between the medium decoupling scenario and the longer or shorter decoupling scenarios. These observations support our hypothesis S3. However, this hypothesis is rejected for the other 7 scenarios for which no difference of relevant size was found.

Table 9: Comparison of tt_{dep} for shorter decoupling duration of 1:30 minutes and longer decoupling duration of 2:30 minutes with respect to the base case of 2:00 minutes. Before the comparison, results were corrected for the input difference in decoupling time. Results for runway 36L (a) and 36C (b). Significance level $0.05/2 = 0.025$

(a) RWY 36L					(b) RWY 36C				
Decoupling 1:30 min		Decoupling 2:30 min			Decoupling 1:30 min		Decoupling 2:30 min		
p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test	
ES Short	< 0.001 (0.42, S)	0.17	-		ES Short	0.11	-	< 0.001 (0.61, S)	
ES Medium	< 0.001 (0.42, S)	0.16	-		ES Medium	0.05	-	< 0.001 (0.63, S)	
ES Long	0.0017 (0.45, N)	0.082	-		ES Long	0.33	-	< 0.001 (0.63, S)	

6.2 Evaluating RMO South

6.2.1 Global Trend Analysis

In the following section, we outline the results for RMO South with respect to our hypotheses and compare to what extent the findings are in line with our observations in the previous section. Table 10 shows the results for RMO South. We first observe that the medians of tt_{dep} are lower than under RMO North, whereas the medians of tt_{arr} are higher. This difference with respect to the results in the previous section is explained by the shorter distance between the departure runways and the ramps, and the longer distance between the arrival runways and the ramps when RMO South is active.

Table 10: Comparison of tug-enabled taxiing (TB) with Multi-Engine Taxiing (MET) with variation of Engine Start duration (ES) and Decoupling Duration. Significance level $0.05/3 = 0.0167$

Short Engine Start: ICAO A, B, C: 2:15 minutes and ICAO D,E,F 4:30 minutes							
KPI	MET	TB1 - Decoupling 1:30 min		TB2 - Decoupling 2:00 min		TB3 - Decoupling 2:30 min	
	median (Q1, Q3)	median (Q1, Q3)	p-value, A-test	median (Q1, Q3)	p-value, A-test	median (Q1, Q3)	p-value, A-test
td_{arr}	7.28 (3.41, 8.41)	7.31 (3.39, 8.41)	0.73, (-)	7.28 (3.39, 8.42)	0.77, (-)	7.29 (3.41, 8.43)	0.82, (-)
td_{dep}	1.78 (1.10, 2.40)	1.92 (1.30, 2.42)	<0.001, (0.55, N)	1.92 (1.30, 2.46)	<0.001, (0.557, N)	1.95 (1.32, 2.49)	<0.001, (0.563, S)
tt_{arr}	9.85 (5.44, 11.09)	9.85 (5.41, 11.09)	0.48, (-)	9.80 (5.43, 11.09)	0.49, (-)	9.83 (5.41, 11.09)	0.56, (-)
tt_{dep}	6.66 (5.27, 8.04)	7.35 (6.12, 8.64)	<0.001, (0.61, S)	7.99 (6.66, 9.28)	<0.001, (0.69, M)	8.62 (7.24, 9.87)	<0.001, (0.76, L)
tt_{BA}	3.15 (2.83, 4.37)	1.68 (1.32, 2.28)	<0.001, (0.088, L)	1.72 (1.32, 2.32)	<0.001, (0.094, L)	1.75 (1.32, 2.29)	<0.001, (0.10, L)
tt_{DC}	N/A	1.50 (1.50, 1.50)	N/A	2.00 (2.00, 2.00)	N/A	2.50 (2.50, 2.50)	N/A

Medium Engine Start: ICAO A, B, C: 3:00 minutes and ICAO D,E,F 6:00 minutes							
KPI	MET	TB1 - Decoupling 1:30 min		TB2 - Decoupling 2:00 min		TB3 - Decoupling 2:30 min	
	median (Q1, Q3)	median (Q1, Q3)	p-value, A-test	median (Q1, Q3)	p-value, A-test	median (Q1, Q3)	p-value, A-test
td_{arr}	7.27 (3.41, 8.41)	7.29 (3.40, 8.41)	0.77, (-)	7.28 (3.41, 8.43)	0.79, (-)	7.29 (3.41, 8.43)	0.86, (-)
td_{dep}	1.78 (1.10, 2.43)	1.92 (1.30, 2.45)	<0.001, (0.55, N)	1.92 (1.32, 2.46)	<0.001, (0.557, N)	1.95 (1.32, 2.49)	<0.001, (0.562, S)
tt_{arr}	9.85 (5.49, 11.09)	9.83 (5.41, 11.09)	0.69, (-)	9.80 (5.48, 11.09)	0.79, (-)	9.83 (5.41, 11.09)	0.74, (-)
tt_{dep}	7.41 (5.95, 9.09)	7.39 (6.22, 8.66)	0.72, (-)	7.96 (6.72, 9.23)	<0.001, (0.58, S)	8.65 (7.27, 9.88)	<0.001, (0.66, M)
tt_{BA}	3.90 (3.58, 5.11)	1.73 (1.32, 2.32)	<0.001, (0.040, L)	1.70 (1.32, 2.28)	<0.001, (0.041, L)	1.75 (1.33, 2.30)	<0.001, (0.050, L)
tt_{DC}	N/A	1.50 (1.50, 1.50)	N/A	2.00 (2.00, 2.00)	N/A	2.50 (2.50, 2.50)	N/A

Long Engine Start: ICAO A, B, C: 3:45 minutes and ICAO D,E,F 7:30 minutes							
KPI	MET	TB1 - Decoupling 1:30 min		TB2 - Decoupling 2:00 min		TB3 - Decoupling 2:30 min	
	median (Q1, Q3)	median (Q1, Q3)	p-value, A-test	median (Q1, Q3)	p-value, A-test	median (Q1, Q3)	p-value, A-test
td_{arr}	7.28 (3.41, 8.40)	7.28 (3.39, 8.41)	0.66, (-)	7.28 (3.39, 8.42)	0.96, (-)	7.28 (3.41, 8.41)	0.93, (-)
td_{dep}	1.77 (1.09, 2.43)	1.92 (1.29, 2.46)	<0.001, (0.55, N)	1.95 (1.32, 2.48)	<0.001, (0.561, S)	1.93 (1.33, 2.50)	<0.001, (0.57, S)
tt_{arr}	9.86 (5.55, 11.09)	9.85 (5.41, 11.09)	0.68, (-)	9.83 (5.43, 11.08)	0.74, (-)	9.81 (5.43, 11.10)	0.93, (-)
tt_{dep}	8.18 (6.78, 9.80)	7.72 (6.18, 8.78)	<0.001, (0.41, S)	8.28 (6.85, 9.37)	0.20, (-)	8.68 (7.28, 9.86)	<0.001, (0.56, N)
tt_{BA}	4.66 (4.33, 5.65)	1.69 (1.32, 2.29)	<0.001, (0.021, L)	1.75 (1.32, 2.32)	<0.001, (0.029, L)	1.75 (1.32, 2.32)	<0.001, (0.027, L)
tt_{DC}	N/A	1.50 (1.50, 1.50)	N/A	2.00 (2.00, 2.00)	N/A	2.50 (2.50, 2.50)	N/A

MET: Multi Engine Taxiing, ES: Engine start duration, td_{dep} : taxi time departures, td_{dep} : taxi distance departures, tt_{arr} : taxi time arrivals, td_{arr} : taxi distance arrivals, tt_{BA} : bay area taxi time, tt_{DC} : decoupling stand-still time. Effect sizes: N: Negligible, S: Small, M: Medium, L: Large. N/A: not applicable.

Table 10 shows a significant difference in tt_{dep} for 7 out of 9 Tug-Enabled Taxiing scenarios with respect to the MET case. At least a small increase is observed in all Tug-Enabled Taxiing scenarios with a short engine start duration. Furthermore, the scenarios with medium engine start duration and medium (TB2) or large decoupling time (TB3) show a small and medium increase, respectively. No significant difference in tt_{dep} is observed for the scenario with 1:30 minutes decoupling time (TB1) and medium engine start duration. In the scenarios with a long engine start duration, no increases in tt_{dep} of considerable size are found. However, we observe a small decrease in tt_{dep} with a difference in medians of 28 seconds for the scenario with long engine start time and short decoupling duration (TB1). This suggests that the use of Tug-Enabled Taxiing in fact leads to a decrease in tt_{dep} . We therefore reject hypothesis G1 for the three scenarios with a long engine start duration and for the scenario with a medium engine start time and short decoupling duration. Furthermore, the absence of a significant difference in arrival taxi times tt_{arr} for the RMO South results strengthens the observation that arrival flights are not significantly influenced by outbound Tug-Enabled Taxiing. These results therefore further support the evidence to reject hypothesis G2.

For increasing engine start durations and increasing decoupling times under RMO South we observe a similar trend as in the previously discussed results, larger effect sizes for the increase in tt_{dep} for longer decoupling and shorter engine start. This provides additional evidence to support hypotheses S1 and S2. The departure taxi distances td_{dep} however, show a different result under RMO South than under RMO North. As can be observed from Table 10, td_{dep} increases with a small effect size in 4 Tug-Enabled Taxiing scenarios. These are all scenarios with a long decoupling duration, and the scenario with a long engine start duration and medium decoupling duration. We further analyze this effect on runway level in section 6.2.2.

On a local level, a large decrease of bay area taxi times for departure flights is observed in all nine Tug-Enabled

Taxiing scenarios. With a decoupling time of 2:00 minutes (TB2), the difference in medians is 1:26, 2:12, and 2:55 minutes for short, medium, and long engine start, respectively. Note that the absolute decrease in median tt_{BA} is comparable to what was previously found for RMO North, within a range of 10s. Even with decoupling locations closer to the ramps, Tug-Enabled Taxiing thus reduces the bay area taxi times, supporting our hypothesis L1.

Figure 9 shows the distribution of tt_{dep} for RMO South. We observe that the share of taxi time in the bay areas with respect to the total taxi time is larger than under RMO North. With similar absolute values for the reduction of tt_{BA} , we can thus state that the reduction in bay area taxi time is more influential under RMO South. Furthermore, the green bars in Figure 9 show that, on average, the difference in other taxi time between MET and the TB scenarios is smaller than under RMO North. This suggests that flights assigned to runways closer to the ramp areas suffer from the slower taxiing velocity of tugs to a lesser extent. This makes sense, as the distance between the ramps and the runway is generally shorter given the layout of Amsterdam Airport Schiphol.

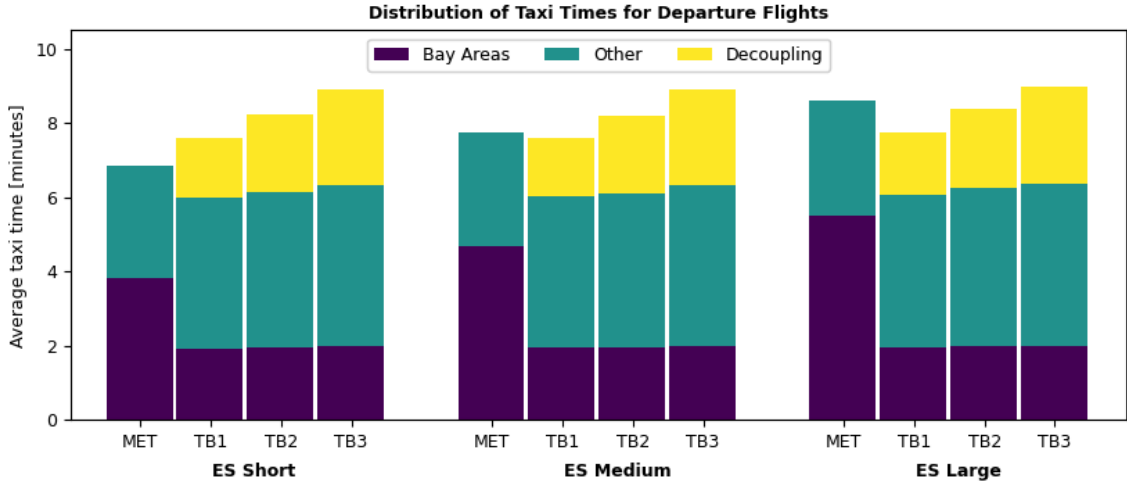


Figure 9: Distribution of outbound taxi time for RMO South. MET: Multi-Engine Taxiing, TB: Tug-Enabled Taxiing. TB1: Decoupling time of 1:30 min, TB2: Decoupling time of 2:00 min, TB3: Decoupling time of 2:30 min.

6.2.2 Runway Specific Analysis: 18L and 24

The results for departure runways 18L and 24 are shown in tables Table 11 and Table 12. For runway 18L, a significant difference in tt_{dep} is observed for 7 out of 9 Tug-Enabled Taxiing scenarios, of which 6 have at least a small effect size. These scenarios are the same as in the global results for RMO South, namely all with short engine start time and two with a medium engine start time and decoupling time of 2:00 minutes or 2:30 minutes (TB2 and TB3). With a long engine start time and decoupling duration of 1:30 minutes, tt_{dep} decreases significantly with a medium effect size and a difference in medians of 53 seconds. For the other two Tug-Enabled Taxiing scenarios with a large engine start time, the differences are not significant or not relevant in size. The observations from the results for runway 18L regarding the research hypotheses are therefore in line with the previously mentioned observations on a global level.

Table 11: Comparison of tug-enabled taxiing (TB) and Multi-Engine Taxiing (MET) for departure flights from RWY 18L. Significance level $0.05/3 = 0.0167$

KPI		MET	TB1 - Decoupling 1:30 min		TB2 - Decoupling 2:00 min		TB3 - Decoupling 2:30 min	
		median	median	p-value, A-test	median	p-value, A-test	median	p-value, A-test
ES Short	td_{dep} [km]	2.30	2.29	0.80, (-)	2.30	0.19, (-)	2.30	0.16, (-)
	tt_{dep} [min]	7.36	7.94	<0.001, (0.64, S)	8.53	<0.001, (0.72, L)	9.00	<0.001, (0.79, L)
	tt_{BA} [min]	3.13	1.78	<0.001, (0.099, L)	1.78	<0.001, (0.11, L)	1.77	<0.001, (0.12, L)
ES Medium	td_{dep} [km]	2.31	2.29	0.96, (-)	2.29	0.27, (-)	2.29	0.12, (-)
	tt_{dep} [min]	8.24	8.07	0.26, (-)	8.49	<0.002, (0.57, S)	9.03	<0.001, (0.66, M)
	tt_{BA} [min]	3.96	1.78	<0.001, (0.041, L)	1.79	<0.001, (0.053, L)	1.77	<0.001, (0.058, L)
ES Long	td_{dep} [km]	2.29	2.29	0.41, (-)	2.29	0.41, (-)	2.29	0.12, (-)
	tt_{dep} [min]	9.02	8.14	<0.001, (0.35, M)	8.64	<0.004, (0.45, N)	9.09	0.03, (-)
	tt_{BA} [min]	4.64	1.75	<0.001, (0.016, L)	1.77	<0.001, (0.029, L)	1.77	<0.001, (0.037, L)

MET: Multi-Engine Taxiing, TB: Tug-Enabled Taxiing, ES: Engine start duration, td_{dep} : taxi distance departures, tt_{dep} : taxi time departures, tt_{BA} : bay area taxi time. ES Short: 2:15 (ICAO A-C) and 4:30 minutes (ICAO D-F), ES Medium: 3:00 and 6:00 minutes, ES Long: 3:45 and 7:00 minutes.

For runway 24, tt_{dep} increases significantly for 6 out of 9 Tug-Enabled Taxiing scenarios as shown in Table 12. As opposed to the results for runway 18L, we observe a small increase in tt_{dep} for the TB3 scenario with a long engine start duration. This suggests that the probability that tt_{dep} increases is larger for runway 24 than for runway 18L for this scenario. Furthermore, the decrease in tt_{dep} for the scenario with a long engine start time and short decoupling duration is found to be larger for runway 18L than for runway 24. This indicates that flights to runway 18L are more likely to experience a shorter tt_{dep} in this scenario.

Furthermore, Table 12 shows a small increase of taxi distance td_{dep} for runway 24 in all scenarios. The absolute difference in medians is again less than 250 meters and can be explained by the slight detours that aircraft have to make to pass a decoupling location before take-off. Heatmaps reported in Appendix C support this observation.

Table 12: Comparison of tug-enabled taxiing (TB) and Multi-Engine Taxiing (MET) for departure flights from RWY 24. Significance level $0.05/3 = 0.0167$

KPI		MET	TB1 - Decoupling 1:30 min		TB2 - Decoupling 2:00 min		TB3 - Decoupling 2:30 min	
		median	median	p-value, A-test	median	p-value, A-test	median	p-value, A-test
ES Short	td_{dep} [km]	1.48	1.65	<0.001, (0.60, S)	1.69	<0.001, (0.60, S)	1.70	<0.001, (0.60, S)
	tt_{dep} [min]	5.84	6.64	<0.001, (0.63, S)	7.33	<0.001, (0.72, L)	8.06	<0.001, (0.78, L)
	tt_{BA} [min]	3.20	1.63	<0.001, (0.083, L)	1.63	<0.001, (0.087, L)	1.73	<0.001, (0.095, L)
ES Medium	td_{dep} [km]	1.43	1.68	<0.001, (0.60, S)	1.72	<0.001, (0.60, S)	1.70	<0.001, (0.61, S)
	tt_{dep} [min]	6.58	6.81	0.098, (-)	7.34	<0.001, (0.61, S)	8.07	<0.001, (0.69, M)
	tt_{BA} [min]	3.90	1.63	<0.001, (0.039, L)	1.63	<0.001, (0.034, L)	1.75	<0.001, (0.044, L)
ES Long	td_{dep} [km]	1.43	1.68	<0.001, (0.60, S)	1.73	<0.001, (0.61, S)	1.72	<0.001, (0.61, S)
	tt_{dep} [min]	7.29	6.83	<0.001, (0.42, S)	7.58	0.70, (-)	8.18	<0.001, (0.59, S)
	tt_{BA} [min]	4.72	1.68	<0.001, (0.023, L)	1.72	<0.001, (0.030, L)	1.75	<0.001, (0.021, L)

MET: Multi-Engine Taxiing, TB: Tug-Enabled Taxiing, ES: Engine start duration, td_{dep} : taxi distance departures, tt_{dep} : taxi time departures, tt_{BA} : bay area taxi time. ES Short: 2:15 (ICAO A-C) and 4:30 minutes (ICAO D-F), ES Medium: 3:00 and 6:00 minutes, ES Long: 3:45 and 7:00 minutes.

To test whether increased decoupling time causes additional congestion under RMO South, we further investigated the difference in tt_{dep} between the Tug-Enabled Taxiing scenarios, as shown in Table 13. For runway 18L, we do not find any significant differences and we thus reject hypothesis S3. For runway 24, significant differences are only found for four short and medium engine start durations, of which only 3 show a small effect size. Hypothesis S3 is thus only supported for these three scenarios under RMO South.

Table 13: Comparison of tt_{dep} for shorter decoupling duration of 1:30 minutes and longer decoupling duration of 2:30 minutes with respect to the base case of 2:00 minutes. Before the comparison, results were corrected for the input difference in decoupling time. Results for runway 18L (a) and 24 (b). Significance level $0.05/2 = 0.025$.

(a) RWY 18L					(b) RWY 24				
	Decoupling 1:30 min		Decoupling 2:30 min			Decoupling 1:30 min		Decoupling 2:30 min	
	p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test
ES Short	0.89	n/a	0.42	n/a	ES Short	< 0.001	(0.42, S)	< 0.001	(0.58, S)
ES Medium	0.48	n/a	0.04	n/a	ES Medium	0.002	(0.45, N)	< 0.001	(0.57, S)
ES Long	0.58	n/a	0.33	n/a	ES Long	0.057	n/a	0.93	n/a

6.3 Evaluation of MET under different Engine Start Durations

In the above mentioned results, we observed that the relative difference in tt_{dep} between Tug-Enabled Taxiing and MET becomes smaller with an increasing engine start time, whereas the relative difference in tt_{BA} becomes larger. We also observed that this is mainly an effect of the increasing taxi times in the MET scenarios. In this section, we investigate to what degree the increase in tt_{dep} is a direct effect of the different input for engine start time in those scenarios. We considered the medium engine start case, with engine start times of 3 minutes and 6 minutes for ICAO A-C and ICAO D-F respectively, as the base scenario. The shorter engine start case and longer engine start case were compared to this base scenario. In this comparison, we corrected for the difference in engine start time by subtracting it from all scenarios. For both input scenarios, no significant difference of relevant size was observed between the groups based on the Wilcoxon signed-rank test and the A-test. Results from these tests are reported in Appendix D. This suggests that engine start duration is the only factor that significantly influences the longer taxi time duration in the MET scenarios.

6.4 Evaluation of Bay Area Taxi Time Reduction

For both simulated days, the previously discussed results showed a reduction of bay area taxi times for the Tug-Enabled Taxiing scenarios with respect to the MET scenarios. An additional analysis was performed to zoom into this difference. To do so, we analyzed the differences in bay area taxi time with respect to the MET case for flights using a push-back maneuver, flights using a push-pull maneuver, and for flights using a taxi-out maneuver. Violin plots of each of the three maneuvers under different engine start durations are shown in Figure 10. It is observed that the reduction in bay area taxi time is the largest for flights that perform a push-pull maneuver, as is also shown in Table 14. This is explained by the fact that those flights are only allowed to start their engines after finishing the entire push-pull maneuver when using MET. When using Tug-Enabled Taxiing, aircraft do not have to await the pull and can directly move using the tug. Aircraft using a taxi-out maneuver do not require a push-back truck when using MET, but have to wait for engine start at their ramp before starting to taxi. Therefore, the reduction in bay area taxi time for taxi-out ramps is largely equal to the engine start time. Finally, we observe the smallest difference for push-back. In the MET scenarios, aircraft are allowed to begin the engine start procedure when they start the push-back. Therefore, the use of Tug-Enabled Taxiing for flights that push-back leads to a lower reduction than for flights using a push-pull or taxi-out.

Table 14: Median bay area taxi time difference δtt_{BA} per push-back type category.

	Short ES			Medium ES			Long ES		
	push-back	taxi-out	push-pull	push-back	taxi-out	push-pull	push-back	taxi-out	push-pull
median δtt_{BA} [min]	-1:28	-2:11	-4:27	-2:13	-2:56	-5:03	-2:57	-3:41	-6:16

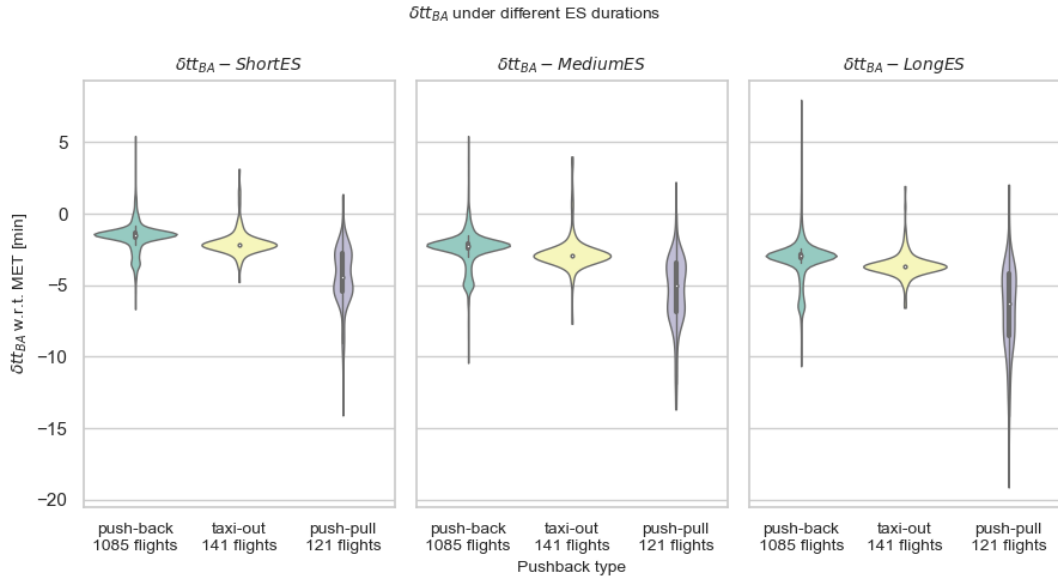


Figure 10: Bay area taxi time differences δtt_{BA} with Tug-Enabled Taxiing w.r.t. Multi-Engine Taxiing split per ICAO group and engine start (ES) time for a decoupling duration of 2:00 minutes.

7 Discussion

The previously discussed results showed how Tug-Enabled Taxiing compares to Multi-Engine Taxiing in the context of Hierarchical Multi-Agent Planning. The interpretation, broader implications, limitations, and reflections on the results and approaches of this study are discussed in this section.

Comparison Tug-Enabled Taxiing and Multi-Engine Taxiing

The results from this study demonstrate that the difference between Tug-Enabled Taxiing and Multi-Engine Taxiing is significantly influenced by engine start time, decoupling duration, and runway configuration. For departure runways located relatively far from the gates, Tug-Enabled Taxiing operations were shown to significantly increase taxi times for departure flights for 8 out of 9 scenarios, thereby partly supporting our hypothesis (G1). Lower taxiing velocity with Tug-Enabled Taxiing and time for tug decoupling are the main causes for this increase. Bay area taxi times reduced in all scenarios, as engine start is performed during taxiing instead of during standstill, which is in line with our hypothesis (L1). When the engine start time is long, and the decoupling duration is short, the increased taxi time due to slower taxiing and decoupling is largely balanced out by this reduction of bay area taxiing time under RMO North. For this reason, we do not observe an increase in total taxi time and reject G1 for this scenario. With longer engine start times and shorter decoupling durations, relative differences in taxi time become smaller which is in line with our hypotheses S1 and S2. For departure runways closer to the gates, under RMO South, we observe smaller relative differences in total taxi time between Tug-Enabled Taxiing and Multi-Engine Taxiing than under RMO North. Only 5 out of 9 scenarios show an increase in outbound taxi times of relevant size, which means that our expectation that outbound taxi times increase only holds for these RMO South scenarios. Whereas the disadvantage of lower maximum taxiing velocity becomes less influential as the taxi distances are shorter, the advantage of engine start during taxiing becomes more apparent. In general, we can thus state that, from a perspective of increasing efficiency, tugs seem favourable when departure runways are close and when engine start time forms a large share of the total taxi time. However, from the consideration of reducing engine-off time, it is questionable whether this forms the best use case. It is therefore suggested to explicitly study engine-on time during taxiing and relate this to emissions in future research.

Variation of engine start time was observed not to have a considerable influence on the Tug-Enabled Taxiing scenarios with respect to each other. The relative differences between Tug-Enabled Taxiing and Multi-Engine Taxiing become larger mainly due to the increased taxi time in the MET scenarios. This result can be of interest with the increasingly longer engine start times needed for newer jet engines [8]. Our results indicate that, if airports want to mitigate further congestion of the bay areas under this new trend, Tug-Enabled Taxiing could be an option to consider. Variation of decoupling time was not shown to result in a large additional increase of taxi times within the Tug-Enabled Taxiing scenarios. Support for the related hypothesis that the taxi time increases by more than the difference in decoupling time (S3) was only observed in 5 of 12 scenarios under RMO North and 3 of 12 scenarios under RMO South. Therefore, we do not observe clear evidence to either support or reject this hypothesis in general. This is in line with a recent study [22] that considered these decoupling points in detail and showed that delays due to decoupling depend on the configuration and capacity of the decoupling locations. Therefore, we recommend to further investigate the effects of decoupling duration and the placements of the decoupling points in future research.

Furthermore, results show that inbound taxi times are not significantly affected in the Tug-Enabled Taxiing scenarios. This means that our hypothesis that inbound taxi times decrease (G2) is rejected for all scenarios. However, outliers were observed in the arrival taxi time data that could indicate the presence of differences on an individual level. Besides that, gate assignment was not considered in this study, so operational context about the reason why aircraft were assigned to their gates remains unavailable. Further research should study the connection between gate availability in more detail.

Finally, our work shows that the type of push-back maneuver influences to what extent bay area taxi times reduce. On an individual level, flights using a push-pull maneuver to leave the ramp experience a higher reduction in bay area taxi time with respect to MET in comparison to flights that use taxi-out or push-back. This observation could be used at airports operating with a limited fleet of tugs, when prioritizing which flights are assigned to use a tug for taxi-out. Whereas a similar observation is made for flights using taxi-out with respect to flights that use push-back, prioritization between these two should be weighed based on operational goals. Flights at taxi-out gates do not block space for other traffic, and therefore mainly benefit on individual basis.

Reflection on Assumptions and Case Study

We analyzed Tug-Enabled Taxiing operations by using an airport layout with a high resolution, and thereby studied the concepts at a high level of detail. This allowed for the evaluation of bay area movements such as

push-back and engine start in the comparison between Tug-Enabled Taxiing and MET, adding valuable insights to previous research. However, several assumptions were made to narrow the scope of this project. First, tug return movements and all-clear operations were omitted from the model. In reality, tugs have to drive to an all-clear point, give the pilot the all-clear sign, and drive back to the tug base after decoupling. This assumption therefore simplifies the processes around the decoupling locations and on the service roads. The inclusion of all-clear operations and tug return movements likely leads to less routing options and more congestion around the decoupling points and is recommended to be included in future iterations of the model. Second, engine start times were categorized in two groups in this research: for small aircraft and larger aircraft. In reality, engine start times are highly variable dependent on manufacturer, outside temperature, starting temperature, and airline procedures. Calibration of the inputs per aircraft type would further increase the realism of the bay area movements. Third, kinematic values for aircraft and tugs in this study are based on estimates. Although the algorithm did account for maximum velocity, acceleration and deceleration rates in planning, these values were kept constant for all aircraft types in this case study. One type of tug was used for both heavy and lighter aircraft, so inclusion of a potential tug variant for wide body aircraft is suggested for future studies. The kinematic inputs to the model could thus be calibrated based on the data of individual aircraft types and tug types. Finally, it is suggested to integrate gate occupancy and holding operations in the model, which is now possible by the inclusion of the ramps. This could help further evaluating the concept of Tug-Enabled Taxiing on a bay area level.

Hierarchical Multi-Agent Planning

Note that this study considered a futuristic scenario in which airport surface movement operations are fully automated using hierarchical multi-agent planning. Although several innovations in the area of Air Traffic Control are exploring automated and decentralized concepts of controlling air traffic [25], actual implementation of the control and coordination mechanisms in this study require further exploration. Furthermore, aircraft were assumed to execute all instructions exactly as instructed. It would therefore be interesting to further explore the performance of the simulation model under the uncertainties of real-life execution.

In general, the adopted routing algorithm was successful in collision-free routing of the scheduled flights in the case study. The activity based search was found to be well suited for planning push-back, engine start, taxiing, and decoupling operations, and it is expected to be usable in future research due to its modular structure. However, some open points remain that require consideration. First, the use of priorities in PBS means that aircraft are always prioritized over each other for an entire route. Although this results in efficient run-time, conflict-by-conflict resolution, as in Conflict-Based Search [26] could be considered to enhance the routing possibilities explored. Besides that, the shapes of aircraft in queues overlapped in some high-traffic scenarios for aircraft with a low velocity. Although this case is rare, it is a limitation of the algorithmic concept and could be improved by anticipating on the other aircraft's shape better when velocities are low. First ideas on this suggestion are reported in Appendix E.

8 Conclusions

In this study, we designed a novel concept of airport taxiing under hierarchical distributed control and used it to compare Multi-Engine Taxiing operations and Tug-Enabled Taxiing operations. We considered aircraft shapes and accounted for kinematics during conflict-free path planning on a high-resolution airport layout. Besides, push-back, engine start, decoupling of tugs, and the adherence to WTC separation constraints during take-off were explicitly included in planning. We studied real-world schedule data for two days, including different departure and arrival runways for each day. Twelve scenarios were simulated for each day: three multi-engine start scenarios with short, medium and long engine start durations and nine tug-enabled taxiing scenarios with an additional variation of the decoupling time in short, medium and long. To the best of the authors' knowledge, this study is the first to compare these novel concepts on this level of detail.

Results showed a significant decrease in bay area taxi times of outbound flights in all Tug-Enabled Taxiing scenarios. With an increase in engine start time, the relative difference in bay area taxi time becomes larger, favouring Tug-Enabled taxiing. Similarly, the reduction in bay area taxi time was found to be larger for ramps where a push-pull maneuver is used than for ramps where a push-back is used. Although it was expected that the reduction of bay area taxi time would result in a noticeable decrease of inbound taxi times, no significant differences were observed. Furthermore, it was shown that differences in outbound taxi times are significantly influenced by input values for engine start, decoupling time, and simulated day of operations. Whereas longer engine start times decrease the difference in outbound taxi time between Tug-Enabled Taxiing and Multi-Engine Taxiing, longer decoupling times increase this difference. Besides that, it was observed that the lower taxiing velocity of Tug-Enabled Taxiing becomes more apparent for runways far away from the ramps. For this reason, outbound taxi time differences between Tug-Enabled Taxiing and Multi-Engine taxiing were observed to be

smaller under the RMO South configuration at AAS than under RMO North. When the distance between gate and runway increases, the relative differences in taxi time between Tug-Enabled Taxiing and MET increases due to the lower taxiing velocity. In general, longer engine start times and short decoupling durations thus reflect the advantages of Tug-Enabled Taxiing more clearly.

This work has provided insights into the comparison between Tug-Enabled Engine-Off Taxiing and Multi-Engine Taxiing, and especially took a perspective of considering time efficiency. It is recommended to evaluate differences in engine-on time in future research, thereby allowing for a comparison in emissions. Furthermore, we suggest to increase the realism of the modelled operations further to increase practical relevance. As a first step, we suggest the inclusion of tug return movements in the model. In addition, variation of aircraft properties such as engine start time and maximum taxiing velocity is advised to obtain further insights in the comparison between the two modes of operations. Furthermore, we suggest to analyze the concepts with different flight schedules, at other airports, and in comparison to actual operations. This will increase the insights both in the practical applicability of hierarchical multi-agent planning for airport ground surface movement operations and in the comparison between Tug-Enabled Taxiing and Multi-Engine Taxiing.

References

- [1] EUROCONTROL, “EUROCONTROL Forecast Update 2021-2027,” tech. rep., EUROCONTROL, oct 2021.
- [2] EUROCONTROL, “European Aviation in 2040, Challenges of Growth,” tech. rep., EUROCONTROL, 2018.
- [3] G. Schürmann, K. Schäfer, C. Jahn, H. Hoffmann, M. Bauerfeind, E. Fleuti, and B. Rappenglück, “The impact of NOx, CO and VOC emissions on the air quality of Zurich airport,” *Atmospheric Environment*, vol. 41, no. 1, pp. 103–118, 2007.
- [4] U. Kesgin, “Aircraft emissions at Turkish airports,” *Energy*, vol. 31, pp. 372–384, feb 2006.
- [5] M. Lukic, P. Giangrande, A. Hebala, S. Nuzzo, and M. Galea, “Review, Challenges, and Future Developments of Electric Taxiing Systems,” *IEEE Transactions on Transportation Electrification*, vol. 5, pp. 1441–1457, dec 2019.
- [6] R. Guo, Y. Zhang, and Q. Wang, “Comparison of emerging ground propulsion systems for electrified aircraft taxi operations,” *Transportation Research Part C: Emerging Technologies*, vol. 44, pp. 98–109, jul 2014.
- [7] TaxiBot, “TaxiBot, The worlds only certified and operational Taxiing alternative.” Available at: <https://www.taxibot-international.com/>. Accessed on: 21-04-2021.
- [8] Schiphol Airport, “Feasibility Study Sustainable Taxiing at Schiphol,” tech. rep., apr 2022.
- [9] B. Benda, “Agent-Based Modelling and Analysis of Non-Autonomous Airport Ground Surface Operations,” nov 2020. Delft University of Technology.
- [10] C. Stergianos, J. Atkin, P. Schittekat, T. E. Nordlander, C. Gerada, and H. Morvan, “The effects of pushback delays on airport ground movement ,” *Journal of Applied Operational Research*, vol. 7, no. 2, pp. 68–79, 2015.
- [11] P. C. Roling, P. Sillekens, and R. Curran, “The effects of Electric Taxi Systems on airport surface congestion,” in *15th AIAA Aviation Technology, Integration, and Operations Conference*, (Dallas, TX), jun 2015.
- [12] W. van Lingen and P. C. Roling, “Modelling the effects of gate planning on apron congestion,” in *AIAA Aviation 2019 Forum*, pp. 1–10, American Institute of Aeronautics and Astronautics Inc, AIAA, 2019.
- [13] H. Udluft, *Decentralization in Air Transportation*. PhD thesis, Delft University of Technology, Delft, dec 2017.
- [14] K. Fines, A. Sharpanskykh, and M. Vert, “Agent-Based Distributed Planning and Coordination for Resilient Airport Surface Movement Operations,” *Aerospace*, vol. 7, p. 48, apr 2020.
- [15] Schiphol, “Schiphol | Reis zoals je wilt reizen via Amsterdam Airport Schiphol,” 2021. Available at: <https://www.schiphol.nl/nl/operations/pagina/sleep-en-pushbackbewegingen/>. Accessed on: 29-04-2021.
- [16] International Civil Aviation Organization, “Annex 14 - Aerodromes - Aerodromes Design and Operations,” tech. rep., International Civil Aviation Organization, 2018.
- [17] W. Honig, J. A. Preiss, T. K. Kumar, G. S. Sukhatme, and N. Ayanian, “Trajectory Planning for Quadrotor Swarms,” *IEEE Transactions on Robotics*, vol. 34, pp. 856–869, 8 2018.
- [18] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. Satish Kumar, E. Boyarski, and R. Barták, “Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks,” in *Proceedings of the Symposium on Combinatorial Search (SoCS)*, pp. 151–158, 2019.
- [19] M. Phillips and M. Likhachev, “SIPP: Safe interval path planning for dynamic environments,” in *Proceedings - IEEE International Conference on Robotics and Automation*, (Shanghai, China), pp. 5628–5635, IEEE, 2011.
- [20] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, “Searching with Consistent Prioritization for Multi-Agent Path Finding,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 7643–7650, 2019.

- [21] R. G. Sargent, “Verification and validation of simulation models,” in *Proceedings of the 2010 Winter Simulation Conference*, (Baltimore, MD, USA), pp. 166–183, IEEE, dec 2010.
- [22] J. Soomers, “A Delay Analysis in Airport Engine-off Towing Operations Using Hierarchical Multi-Agent Path Planning,” jun 2022. Delft University of Technology.
- [23] S. S. Shapiro and M. Wilk, “An Analysis of Variance Test for Normality (Complete Samples),” *Biometrika*, vol. 52, pp. 591–611, dec 1965.
- [24] 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, p. 132, 2000.
- [25] SESAR 2020 Scientific Committee Automation Taskforce, “Automation in air traffic management,” tech. rep., SESAR Joint Undertaking, 2020.
- [26] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, “Conflict-based search for optimal multi-agent pathfinding,” in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, vol. 219, pp. 40–66, feb 2015.

II

Literature Study
previously graded under AE4020

Introduction

The aviation industry has been growing rapidly the past years. In 2018, EUROCONTROL published its forecast for European aviation traffic in 2040 with a predicted growth of 53% in Instrument Flight Rules (IFR) flights with respect to 2017 [23]. Although the effects of COVID-19 have slowed down this predicted growth significantly and even decreased global air traffic numbers, EUROCONTROL considers the scenario that air traffic in Europe will have recovered to 95% of 2019 levels in 2025 as the most likely [25]. At the same time, the importance of sustainability in the aviation sector is rising. Large airports such as Amsterdam Airport Schiphol (AAS) have the vision to be a zero-emission airport by 2030 [63]. Under the assumption that air traffic will continue to grow after full recovery, airports will therefore play an important role in accommodating these increased traffic numbers while at the same time reduce emissions and ensure a safe operation.

Although one way to create additional capacity is to build additional infrastructure, this is a costly investment which, at the same time, makes the airport infrastructure more complex. Besides that, airports can be restricted in their abilities to expand due to space constraints or for environmental reasons. Another more realistic approach is therefore to explore ways to use the existing infrastructure more efficiently. In the conventional operation, Air Traffic Control Officers (ATCO's) are currently responsible for guiding vehicles around the airport safely and efficiently. However, their workload is high and the number of flights they can handle is limited. They therefore rely heavily on procedures that prevent the airport to being used at full capacity.

As another way to control traffic on the airport, a multi-agent distributed approach to control traffic at the airport was proposed in previous research by Fines [29] and Noortman [55]. Various cooperative coordination techniques were tested and were shown to be effective in controlling traffic over the airport. However, a simplification in this research was that gates and aprons have been excluded from the airport layout that was studied. This made it impossible to study local and system wide effects of operations on or around the apron area. This MSc thesis will continue on this research by explicitly focusing on the operations related to the aprons.

In order to make the airport ground surface operation more sustainable, the Advanced Engine Off Navigation (AEON) project, aims to develop a concept for new techniques able to achieve more environmental friendly taxi operations [2]. A promising concept is TaxiBot, an external towing vehicle enabling aircraft to taxi without starting their own engines. It is claimed to lead to 85% fuel savings on the ground and reduced apron blockage times after push-back [96]. Although preliminary studies have been conducted on the system wide effects of TaxiBot implementation at a large airport, apron operations have not been considered before in the context of a multi-agent distributed control system. This MSc thesis will therefore offer the basis for evaluation of this concept at the aprons.

The aim of this thesis is thus twofold. Firstly, an extended airport layout that includes gates and aprons under control of an distributed multi-agent control system will be evaluated. This will make the concept more realistic and the results of this study will therefore provide more detailed insights into the suitability of distributed control mechanisms in the context of airport ground surface operations. Secondly, the effects of the implementation of engine-off taxiing with autonomous TaxiBots on this more realistic airport layout will be

analyzed, with a specific focus on the operations that take place on or around the apron area. The results of this analysis will provide detailed insight in the implications of the implementation of TaxiBots at a large airport and will provide a comparison between the conventional apron operation and a novel concept in which autonomous TaxiBots are used. Amsterdam Airport Schiphol will be used as a case study for this thesis as it representative for a large and busy airport.

Report Structure

From this point onward, this literature study is structured as follows. Conventional operations at Amsterdam Airport Schiphol will be discussed in chapter 2. This chapter will introduce ground surface movement operations and introduce Amsterdam Airport Schiphol. In chapter 3, developments in airport ground surface operations will be discussed. State-of-the-art innovations in ATC support systems, alternative engine-off taxiing methods and new traffic management techniques will be described. In chapter 4, techniques for modelling the ground surface operations will be reviewed and the contributions and shortcomings of previous research projects in the area of agent based modelling of the airport ground surface operation will be evaluated. In chapter 5, multi-agent planning & scheduling will be discussed and different cooperative coordination mechanisms will be reviewed. After discussing the more classical approaches, motion-based approaches to path planning will be discussed. In chapter 6, the research proposal will be outlined including the research objective, research questions and a research planning.

A brief overview of the structure of the report is shown in [Figure 1.1](#) including the purpose of each chapter.

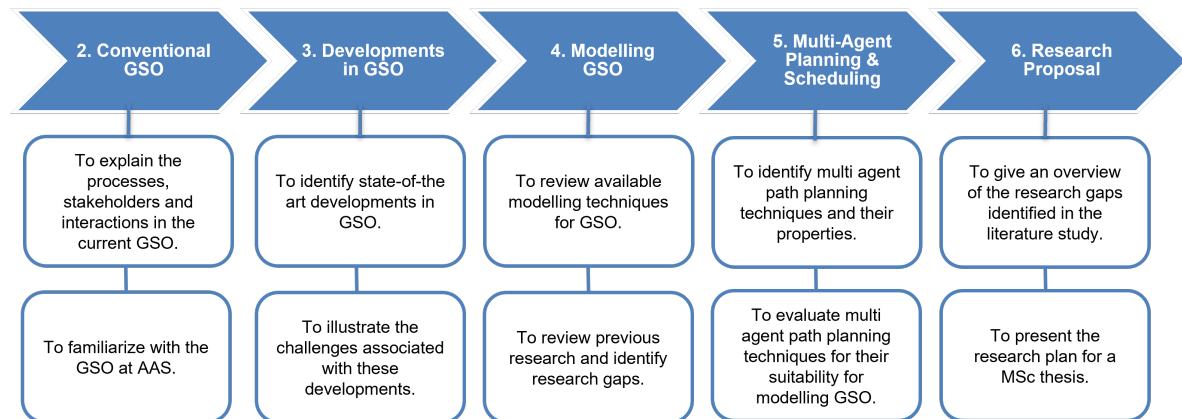


Figure 1.1: Overview of the chapters in this literature study and the purpose(s) of each chapter. GSO refers to Ground Surface Operations.

2

Conventional Ground Surface Operations

This first chapter will cover the conventional ground surface operations at an airport. It is important to understand the processes that are part of the ground surface operations, the different stakeholders and the interactions between them as background for this study. First a general overview of ground surface operations at an airport will be given. Key components of the operation will be discussed as well as the Airport Collaborative Decision Making (A-CDM) system. The milestone approach which is part of the A-CDM will be discussed in more detail as it forms an important basis for understanding the timing of the operations. Communications and interactions between the most important parties at an airport will then be described.

Although there are many similarities between airports, it was chosen to use Amsterdam Airport Schiphol (AAS) as a case study in this study. The airport is found suitable to be used as a case study as it is a relatively large airport in terms of both airport layout and passenger flows. The second part of this chapter will therefore zoom in into Amsterdam Airport Schiphol and discuss the elements of the airport ground surface operation at this airport specifically. In the last section of this chapter, traffic management techniques that are currently used to prevent bottlenecks and mitigate delays at the airport will be discussed.

2.1. General Overview of the Ground Surface Operation at an Airport

In this section, airport ground surface operations will be discussed on a general level. Different stages of the operation will be described, the Airport Collaborative Decision Making (A-CDM) system will be introduced and the communications and interactions in the airport ground surface operation will be outlined.

2.1.1. Key Components of the Ground Surface Operation

The different stages of the airport ground surface operation are visualized in [Figure 2.1](#), starting from the arrival of an aircraft at the airport in the left top of the image. After arrival, it navigates through the taxiway network towards its assigned aircraft stand/gate. The turn around process then starts and once this is finished, the aircraft is pushed back from the gate onto the apron so that it can taxi to its runway and take-off again. As seen in [Figure 2.1](#), the the airport's infrastructure used for the ground surface operation can be roughly split into three main areas: the runways, the taxiways and the aprons with the aircraft stands. The processes that take place during each of the stages will be described in more detail in [section 2.2](#) by using Amsterdam Airport Schiphol (AAS) as a case study.

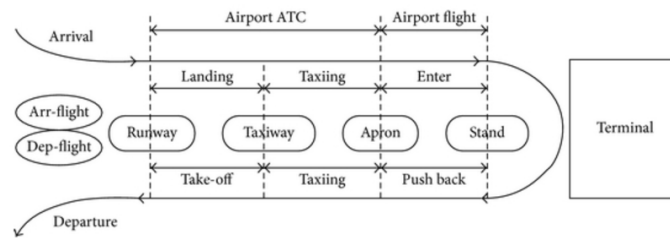


Figure 2.1: Overview of the different stages in the ground surface operation, obtained from Jiang et al. [42].

The milestone approach of the Airport Collaborative Decision Making (A-CDM) system is used to time the most important events during the stages of the ground surface operation. These milestones will be introduced in the next section, after discussing the general A-CDM system.

2.1.2. Airport Collaborative Decision Making (A-CDM)

Many stakeholders are involved in the airport operations process with all their own piece of information about the operation. Sharing more information than strictly necessary can improve the efficiency and predictability of the operation and can therefore be beneficial for many of the parties involved. This is why A-CDM was introduced with the aim to "improve the efficiency and resilience of airport operations by optimising the use of resources and improving the predictability of air traffic" [22]. Amsterdam Airport Schiphol became a CDM airport in 2018 after 27 other European airport had already implemented the system before [67]. Currently 30 European airports, including the largest European hubs, are connected to the CDM network [22]. In this way, the most important stakeholder in the ground operation can share operational data in one integrated network. In this section the Airport Collaborative Decision Making (A-CDM) system will be described in more detail including its implementation at Amsterdam Airport Schiphol. Parts of this system form the basis for the description and analysis of the ground surface operation that will be described later in this chapter.

Key Elements of A-CDM

The key elements of A-CDM are described in the Aircraft Collaborative Decision Making implementation manual by Eurocontrol [26] and are the following: information sharing, the milestone approach, variable taxi time, pre-departure sequencing, adverse conditions elements and collaborative management of flight updates. Each of these elements is first explained below, in line with the description by EUROCONTROL [26] and Schiphol in its own A-CDM manual [77]. The most relevant elements for this research will then be discussed in more detail.

1. **Information Sharing:** The information sharing element of A-CDM prescribes that the most important actors involved in the operation share information accurately and timely. This information can then be used by involved parties so that they are well aware of the current situation and so that future events can be better predicted. At AAS, the Airport Central Information System Schiphol (CISS) is used as a central platform on which all information is shared. Information from separate sources such as ATC, airlines and ground handlers is all linked to this platform.
2. **Milestone Approach:** The aircraft operation can be split in different stages, with different stakeholders in each of the stages. The most important events in the operation are categorized as CDM milestones that occur after each other in the so-called CDM Milestone Approach. The timings of these milestones forms the basis for the analysis of airport ground surface operations and will therefore be discussed more extensively later in this section.
3. **Variable Taxi Time:** The taxi time of an aircraft is the time between the landing and the in block time for arriving flights and the time between the take-off and the off block time for departing aircraft. A common method to determine the standard taxi time is to take a standard and static taxi time for aircraft taxiing from or to the runway that is active at that moment [26]. This standard time does not account for uncertainties such as weather conditions, differences in aircraft type or differences in stand location. In the concept of variable taxi times, the taxi time does incorporate these influences making the taxi time variable and thus more dynamic than a standard value. The variable taxi time is used in order to calculate the estimated in block time (EIBT) for arriving aircraft. For departing aircraft, ATC at Schiphol uses

the variable taxi time to calculate the Target Start-Up Approval Time (TSAT) and the Target Take-Off Time (TTOT) [77].

4. **Pre-Departure Sequencing:** Pre-departure sequencing refers to a process of sequencing the departures of aircraft from the aircraft stand. This is determining a sequence of times that aircraft are cleared for push-back from their stand (or taxi-out in case no push-back is required). A non-optimal process that is commonly used is sequencing based on the first come first serve basis, where aircraft are pushed back in the order that they request permission. This process often results in long queues of aircraft at the runway entry points which are undesirable because of the high waiting time and increased fuel consumption [26]. The idea of pre-departure sequencing is that it leads to less waiting times at the runway points by sequencing aircraft more efficiently before push-back from the stand. For example, it gives ATC the possibility to hold and thus delay aircraft at the gate with the result that the overall situation can be improved. At AAS, the collaborative pre-departure sequence planning system (CPDSP) is used in order to determine a sequence in which optimality and preferences of different stakeholders are taken into account. CPDSP will be discussed more elaborately in [subsection 2.3.1](#)
5. **Adverse Conditions Elements:** Disruptions can occur at airports, influencing the operation of departing and arriving aircraft. Although some causes of disruptions can not be predicted, for example an aircraft that blocks a runway because it crashed, there are also events that can be predicted well. EUROCONTROL identified the most significant predictable adverse conditions to be weather, construction or maintenance work at the airport and the (un)availability of technical resources [26]. For the remainder of this research, it is assumed that normal operations occur at the airport. The relevance of the adverse conditions in A-CDM is therefore limited in this thesis and will not be elaborated on any further.
6. **Collaborative Management of Flight Updates:** The collaborative management of flight updates refers to a process of information sharing between an A-CDM airport and the EUROCONTROL Network Manager Operations Centre (NMOC). At AAS it is used to exchange information about the expected landing times of arriving aircraft and the expected departure times of departing aircraft. As aircraft are assumed to land at a scheduled arrival time in this research, the relevance of this last element is limited and will therefore not be discussed further in the remainder of this thesis.

Having introduced the six elements of A-CDM in relation to the operation at Schiphol, the most relevant elements of A-CDM will be discussed in more detail. Those are the milestones that are part of the milestone approach (discussed below) and the collaborative pre-departure sequence planning (CPDSP) system (discussed in [subsection 2.3.1](#)) that is part of the pre-departure sequencing element.

The milestone approach

Enabled by the information sharing element of A-CDM, the milestone approach enables stakeholders to monitor the most important events at the airport. The aim is that all stakeholders have common situational awareness and that future events can be predicted for each flight [26]. The sequence of the different milestones is shown in [Figure 2.2](#). These will be further discussed in this subsection in line with the description of these milestones at AAS as described in the Schiphol A-CDM manual [77] and the EUROCONTROL manual [26]. On a high level, three phases can be distinguished: inbound, turn around and outbound.

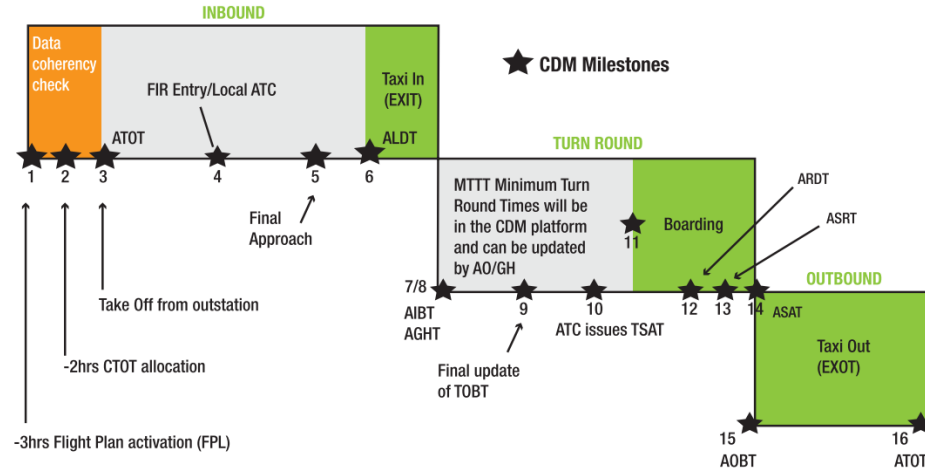


Figure 2.2: Overview of the CDM Milestones [26].

The first phase is the inbound phase marked by the different milestones between the departure from the outstation towards the parking of the aircraft at the aircraft stand at the A-CDM airport. The most relevant milestones in the area of ground surface operations in this stage are milestone 6 and 7, described in Table 2.1.

Table 2.1: Two most relevant milestones of the A-CDM inbound phase

Milestone 6	Aircraft landing: An arriving aircraft has landed at the CDM airport. Actual time of landing is saved to the CDM system as the actual landing time (ALDT). After this milestone, the aircraft enters the taxiway network and starts taxiing to its designated stand. Estimated taxi-in time is saved as EXIT.
Milestone 7	Aircraft parked at stand: An arriving aircraft arrives at its designated stand after taxiing. The arrival time at the stand is saved as the actual in block time (AIBT). Actual taxiing time is saved as AXIT and is equal to AIBT-ALDT.

The second phase in A-CDM is the turn around phase, which consists of milestones that take place at the aircraft stand of the A-CDM airport. These are described in Table 2.2.

Table 2.2: The milestones of the A-CDM turn around phase

Milestone 8	Ground handling starts: The ground handling of an arriving aircraft starts shortly after it has arrived at the stand. At AAS, milestone 7 and 8 are therefore merged.
Milestone 9	Final update of TOBT: The earliest possible target off-block time (TOBT) is calculated automatically based on the minimum turn round times (MTTT). This is done by adding the MTTT to the AIBT and represents the earliest time an aircraft is theoretically able to vacate the stand again. The TOBT can and should be updated by the ground handler or the aircraft if there is an indication that the original TOBT is not achievable. AAS asks to evaluate the achievability of the TOBT at an accuracy of ± 10 minutes and update the TOBT if this is not achievable [77]. Milestone 9 represents the moment that the final update of the TOBT is made.
Milestone 10	ATC issues TSAT: Pilots are informed by ATC about the target start up time (TSAT) that has been assigned to them. At AAS, this is usually done via the A-VDGS display which is visible from the cockpit. A variation on the TSAT is the TSAT window which is defined at the TSAT ± 5 minutes at AAS.
Milestone 11	Boarding starts: At this milestone, the boarding process starts.
Milestone 12	Aircraft ready: Once all of the aircraft's doors are closed, the boarding bridge is removed from the aircraft and the push-back vehicle is attached to the aircraft, the aircraft is ready [26]. The time at which this is the case is the actual ready time (ARDT).
Milestone 13	Aircraft requests start up: Once the aircraft is ready, it requests clearance from ATC to start up. This milestone is defined by the actual start up request time (ASRT).
Milestone 14	Start up approval: The time at which start up approval is given to the aircraft by ATC is the actual start up approval time (ASAT).

The final phase of the A-CDM process is the outbound phase where the aircraft leaves the CDM airport again. The milestones in this phase are described in Table 2.3.

Table 2.3: The milestones of the A-CDM outbound phase

Milestone 15	Aircraft leaves stand: Shortly after start up approval has been received by the aircraft it leaves the aircraft stand and starts to taxi to its designated runway. The time at which the stand is vacated is the actual off-block time (AOBT). From this point, the estimated taxi-out time (EXOT) is the time the aircraft is predicted to need to taxi to the runway.
Milestone 16	Take-off: After the aircraft has taxied from the stand to the runway the aircraft takes off. This time is represented by the actual take-off time (ATOT). The time the aircraft needed to taxi from the gate to the runway is the actual taxi-out time (AXOT) and is calculated by ATOT-AOBT.

As shown in the subsection above, the milestone approach provides stakeholders information about the most important events at the airport during the operation. In [subsection 2.1.3](#), the milestones will be used in order to describe the communications and interactions between these different stakeholders. Later in this research, the historic milestone data can be used to compare modelled operations with the timings of the actual operation.

2.1.3. Communications and Interactions

In the description of the A-CDM system it has become apparent that many different stakeholders are involved. The aim of this section is to give an overview of the most important actors involved in the ground operation, their responsibility and their interaction with other stakeholders. This overview will be summarized in a socio-technical description of the system of airport ground surface operations. Although the high-level process is similar for all airports, some processes have been specified to be representative for Amsterdam Airport Schiphol already, as this airport will be used as case study in this research project.

Extensive descriptions of the involved parties and their interactions in the airport ground surface operations have been given in past MSc thesis at ATO by Benda [9] and Fines [29]. These were all based on a description of the system by Noortman in his MSc thesis [55]. The descriptions given in these previous studies will be used as the basis for this subsection with an extension of the interaction between the ground handler and the aircraft. This is where the interaction between towing / push-back truck driver is part of.

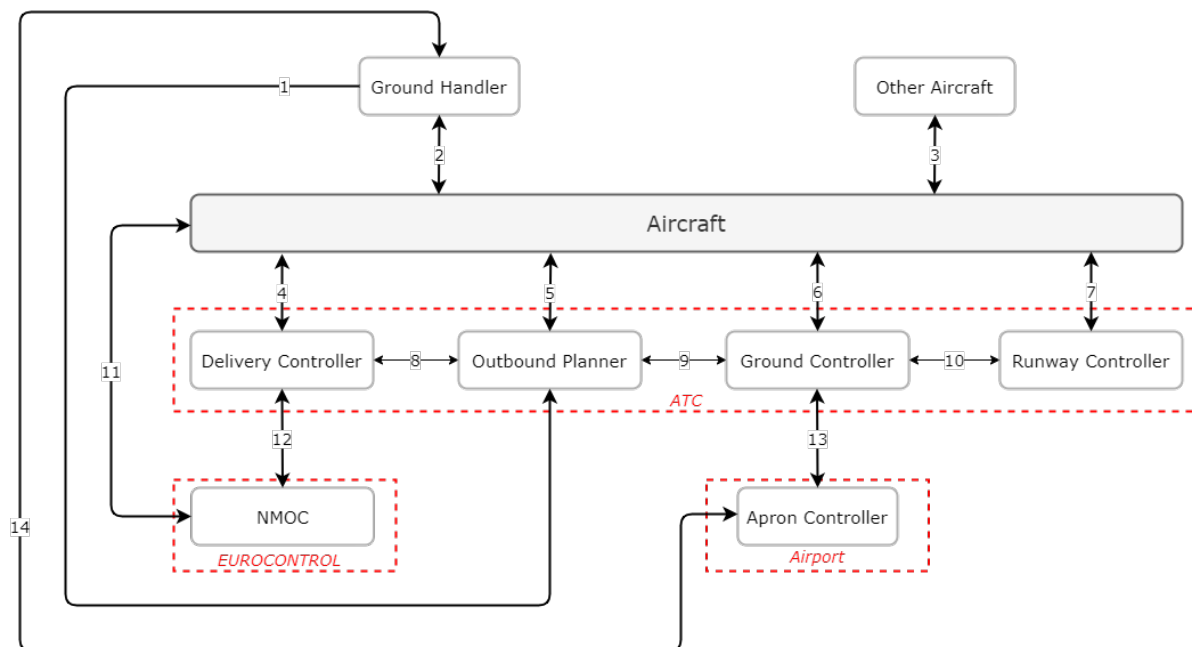


Figure 2.3: Socio-technical system. Own work based on Fines [9] and Noortman [55].

Aircraft + Flight crew

All parties discussed in this overview are related to the aircraft, which refers to the aircraft controlled by a flight crew. The pilot in command has the final responsibility over the aircraft at all times during the aircraft its

flight phases, starting from the moment where the aircraft leaves the gate with a push-back manoeuvre [72]. During or just before the movement on the airport's ground surface, the pilot is constantly in contact with different ATC parties, shown by [links 4,5,6,7](#). A departing aircraft can be restricted in its departure time by the EUROCONTROL network manager, which will then communicate a CTOT to the aircraft and the delivery controller. More detailed, the links in [Figure 2.3](#) connected to the aircraft are:

- **Link 2: Ground Handler ↔ Aircraft:** Communication of adjustment of TOBT by ground handler to the aircraft. A-VDGS is used at aircraft stand to indicate whether arriving aircraft are allowed to enter stand. The tug driver, who does the push-back of the aircraft, interacts with the aircraft about this procedure. This is further specified in [Figure 2.10](#) in [subsection 2.2.3](#).
- **Link 3: Other aircraft ↔ Aircraft:** Maintain visual separation.
- **Link 4: Delivery Controller ↔ Aircraft:** Flight plan approval by delivery controller and issuance of TSAT. Confirmation of TSAT by Aircraft through read back.
- **Link 5: Outbound Planner ↔ Aircraft:** Aircraft lets outbound planner know that it is ready. Depending on the timing with respect to the TSAT window, the outbound planner clears the aircraft for start-up or, in case push-back is required, it orders the aircraft to contact ground control.
- **Link 6: Ground Controller ↔ Aircraft:** Aircraft contacts ground control after being instructed to do so by outbound planner. Push-back clearance is requested at ground controller by aircraft. Ground controller gives push-back clearance and then taxi clearance. Arriving aircraft contact ground control after vacating the runway and receive taxi instructions. Aircraft crew reads back instructions.
- **Link 7: Runway Controller ↔ Aircraft:** Aircraft contacts runway controller for take-off or landing clearance. Runway controller gives clearance to aircraft.
- **Link 11: NMOC ↔ Aircraft:** NMOC communicates CTOT to aircraft, although mostly via delivery controller. Aircraft submits flight plan to NMOC.

Ground Handler

The ground handler provides a set of different services to the aircraft in the turn around process. Examples are cleaning, refueling, loading and unloading. Communication about the status of these processes takes place between the pilot and ground handler when the aircraft is at the stand. Also the push-back service provided to departing aircraft is part of the ground handlers tasks. During push-back the driver of the push-back truck and the pilot follow the standard push-back procedure as will be described in [Figure 2.10](#) in [subsection 2.2.3](#). The main ground handler is responsible for issuing and updating a target off-block time (TOBT) and ensuring that this time is reached. This should be communicated both to the aircraft and the outbound planner. The accuracy of the issued TOBT is expected to be +/- 5 minutes and should be updated by the ground handler if this time is not reached [77]. More detailed, the links in [Figure 2.3](#) connected to the ground handler are:

- **Link 1: Ground Handler → Outbound Planner:** Ground handler updates and communicates TOBT to outbound planner through A-CDM system.
- **Link 2: Ground Handler ↔ Aircraft:** As discussed under aircraft.
- **Link 14: Ground Handler ↔ Apron Controller:** Apron Controller guides towed aircraft over the airport and communicates routing commands to the push-back driver (who is part of the ground handler).

Air Traffic Control (ATC)

The main responsibility of Air Traffic Control in the system is to ensure safety. Besides that, ATC controllers also play an important role in providing an efficient system. In the context of ground surface operations, four relevant different sub parties can be identified within ATC, being the delivery controller, outbound planner, ground controller and runway controller. Interactions between the parties consist of transferring the flight strip ([links 8,9,10](#)). Other interactions will be further elaborated below.

- **Delivery Controller** The main task of the delivery controller is to approve the flight plan submitted by the aircraft. An submitted flight plan includes, among other things, the runway the aircraft is assigned to and the standard instrument departure (SID) the aircraft should follow after take-off. If a calculated take off time (CTOT) is issued by EUROCONTROL network operations, this is communicated to the aircraft by the delivery controller ([Link 12](#)). The delivery controller also interacts with the aircraft in [Link 4](#) as specified under aircraft.

- **Outbound Planner** The outbound planner determines a sequence in which the aircraft can start up or be pushed back from the gate and provides a target start approval time (TSAT) to indicate when the aircraft can expect to receive clearance to do this. If an aircraft is actually ready to leave its stand¹, it lets the outbound planner know ([Link 5](#)). The outbound planner then checks if the aircraft is already in the TSAT-window that was provided. From this point there are three options [77]. If the aircraft is ready before the TSAT-window, it will be asked to wait and call again later in the TSAT-window. If the aircraft is ready after the TSAT-window, a new TSAT should be determined based on a new TOBT. The last option is that the aircraft is ready within the TSAT-window. In this case, the outbound planner will give clearance to start-up directly or wait for a couple of minutes if the taxiway network or runway is too busy. As described for the ground handler, TOBT updates are also communicated to the outbound planner ([Link 1](#)).
- **Ground Controller** The ground controller guides the aircraft over the airport while it is in the taxiway network by giving taxi clearances ([Link 6](#)). For departing aircraft this is from the moment the aircraft passes the red boundary line at the stand till the moment that it is handed over to the runway controller at the runway entry. For arrival aircraft this is the other way around. The ground controller is thus responsible for issuing push-back and taxi clearance. The ground controller can decide to prioritize certain departing aircraft in the taxiway network so that they can adhere to their pre-determined sequence and, if issued, CTOT. Next to guiding aircraft, ground controllers also determine the routes of ground vehicles such as push-back trucks and bird controllers that move on the airport and communicate with apron control about towed aircraft ([Link 13](#)).
- **Runway Controller** The runway controller is responsible for the processes on and around the runway. This controller issues take-off clearances for departing aircraft, taking into account any restrictions such as wake turbulence separation ([Link 7](#)). Landing aircraft require a landing clearance from the runway controller who also takes into account a certain separation distance between arriving aircraft. Finally, aircraft and vehicles that have to move over the runway to cross or for an inspection require clearance from the responsible runway controller.

Network Manager Operations Centre (NMOC)

EUROCONTROL aims to optimize the traffic flow in the European air traffic network. Information about the aircraft, including a required flight plan, is shared with the network manager operations centre through the A-CDM network. As described in [subsection 2.1.2](#), many large European airports are connected to the A-CDM network. Based on this information, and the density of air traffic above Europe, the NMOC can issue an calculated take off time (CTOT) to a departing aircraft so that the overall network situation is improved ([Link 11](#) and ([Link 2](#)).

Apron Controller

While ATC is responsible for the guidance of most ground traffic over the airport, part of this task is delegated to an organization called Apron Control as part of Amsterdam Airport Schiphol. Apron control is responsible for the guidance of towing traffic and non-aircraft vehicles over the airport if visibility conditions are good ([Link 14](#)). Apron control also guides vehicles over service roads and over roads around the apron. Apron controllers listen to the commands given to aircraft by ATC ground control ([Link 13](#)), so that they can take this into account in their instructions to vehicles and towing traffic.

2.2. Ground Surface Operations at Amsterdam Airport Schiphol

After having described the airport ground surface operations in general, the operations at Amsterdam Airport Schiphol (AAS) will be described in this section in more detail. In 1916, Schiphol started as a military airport Haarlemmermeer and consisted of a simple piece of farmland [73]. In the early years after World War I, the airport was also used for transporting passengers and goods. During World War II, the airport was bombed and rebuild multiple times. It was after this war in 1945, when the airport was rebuild for the last time and the Dutch Government declared the airport as the "world airport of the Netherlands" due to its potential for growth as compared to other airports in the Netherlands [73]. From this point onward, the airport grew to one of the largest European airports with around 500,000 air transport movements and over 70 million passengers in 2019 [68]. Due to the high traffic numbers and relatively large size of the airport, AAS is considered

¹An aircraft is considered fully ready when all ground handling processes are finished and when it is ready for immediate push-back or taxi[77].

as a suitable airport to be used as a case study for this research.

In subsection 2.1.1, the stages of the airport ground surface operation were discussed. It was shown that the runway, taxiway system and the aprons and gates are key areas where the airport ground surface movement operation takes place. For reasons that will become apparent later in this report, an additional area that will be considered as a separate part of the airport's infrastructure are the remote holding positions and service roads. The runways, taxiways, aprons (including gates), remote holding positions and service roads will be discussed in the next sections. The most relevant processes, involved actors and their interactions will be described. The aim of this section is to provide required background information about the airport ground surface movement operation by specifically considering Amsterdam Airport Schiphol.

2.2.1. Runways

Amsterdam Airport Schiphol (AAS) has six runways used for take-offs and landings. Their general specifications are shown in Figure 2.4. The Schiphol East runway, 04/22, is generally only used for general aviation and therefore not considered as one of the main runways. In this section, the usage of the five main runways will be described together with the most important factors influencing runway capacity. It is important to understand the runway usage for the remainder of this report, as this forms the interface between the airport operation and the operation that takes place out of the control of the airport.

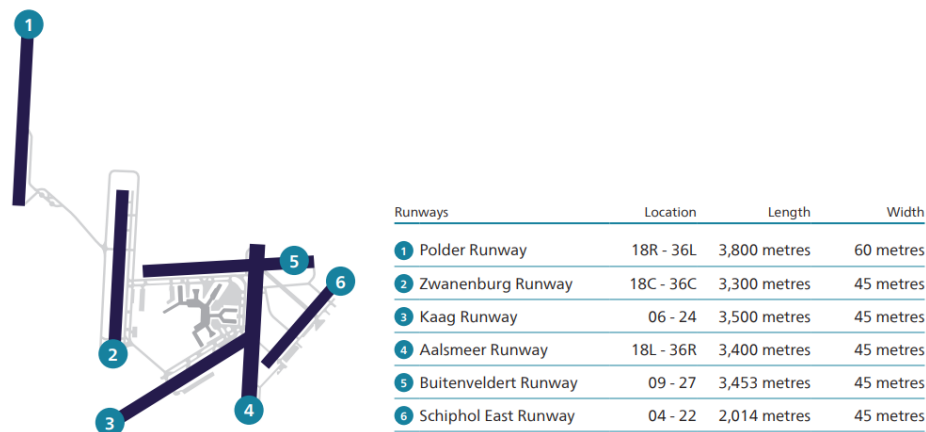


Figure 2.4: Overview of the runways at AAS [68].

The runway usage at Schiphol is based on a set of rules in order to minimize the nuisance for the local residents around the airport. These rules are defined in an annual Runway Usage forecast by Schiphol [10]. One of the restrictions is that the Polderbaan (18R - 36L) can only be used for departures in the Northern direction and for arrivals in the Southern direction. Besides that, the Aalsmeerbaan (18L - 36R) can only be used for departures in the Southern direction and for arrivals in the Northern direction. Next to these restrictions, a preferred sequence is defined with 8 different combinations of runways during daytime and 4 different combinations of runways during night time, shown in Table 2.4. For each combination, 1 or 2 landing runways are defined as well as 1 or 2 runways for departing traffic. When there is not many traffic there are usually 2 active runways, one for departures and one for arrivals. A third runway can be added in departure or arrival peaks. Factors that influence the active runway configuration are for example wind speed and direction, visibility conditions, runway maintenance or emergency situations. In order to use the active runways to their maximal capacity, Schiphol uses the so-called rapid exit taxiways that enable aircraft to vacate the runway quickly after landing. The placement of these exits at 30° relative to the active runway enables aircraft to leave the runway at higher speed so that ATC can use the minimal separation distance between arriving aircraft [49].

Table 2.4: Preferred combinations of landing and take-off runways from 6:00 till 23:00 (a) and from 23:00-6:00(b) [10]. Preferred sequence 1 indicates the first preference, with the next preferences following.

Preferred sequence	Landing 1	Landing 2	Take-off 1	Take-off 2
1	06	36R	36L	36C
2	18R	18C	24	18L
3	06	36R	09	36L
4	27	18R	24	18L
5a	36R	36C	36L	36C
5b	18R	18C	18L	18C
6a	36R	36C	36L	09
6b	18R	18C	18L	24

Preferred sequence	Landing 1	Take-off
1	06	36L
2	18R	24
3	36C	36L
4	18R	18C

In order to get an impression of the runways that are used most at Amsterdam Airport Schiphol, the reported number of commercial take-offs and landings in 2019 was analysed. The percentage of total take-offs and landings on each runway is shown in Figure 2.5. Note that the 2019 data was considered to be more representative for a regular operational year at Schiphol than 2020 due to the reduced number of traffic resulting from COVID-19. It is observed in Figure 2.5 that runway 18R was used for around 40% of the total landings, runway 18C and 06 are both used for around 15% of the total landings, runway 36R for around 13% of total landings and finally runway 36C, runway 27 and runway 22 is used for the remainder of the total landings. It can also be seen that take-offs do most often take place from runway 24 (around 32%), with runways 36L and 18L on a close second place with around 25% of total take-offs each. The other take-offs mainly take place from runway 36C, runway 09 and runway 18C. As will become apparent in the end of this study, a model will be developed that represents AAS in which the information about the runway usage is required for an accurate representation of the airport.

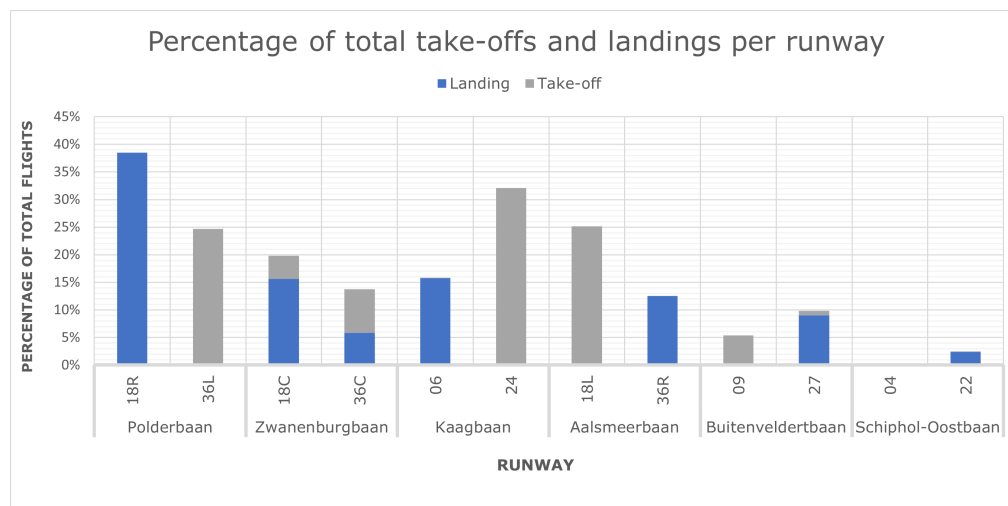


Figure 2.5: Percentage of total take-offs and landings, specified per runway. Own work based on "Baangebruikcijfers" [1].

2.2.2. Taxiway network

The taxiway system is the part of the airport that forms the connection between the aprons and the runways and is controlled by Air Traffic Control. It is of major importance in the ground operation, as all landing and departing traffic uses it to move between the gates and the runways. In the core of the taxiway network at Amsterdam Airport Schiphol, there is a circular taxiway system around the passenger terminals. This circular system is shown in Figure 2.6 and connects all passenger gates to most runways except the the Polderbaan (18R / 36L) and the Schiphol East Runway (04/22). Additional taxiways are connected to the central circular taxiway system to form a connection to runway 18R/36L and to the cargo and maintenance areas. The circular taxiway is connected to runway 04/22 mainly for towing purposes, as the traffic landing and departing from this runway uses the separate K-apron on the other side of the airport.

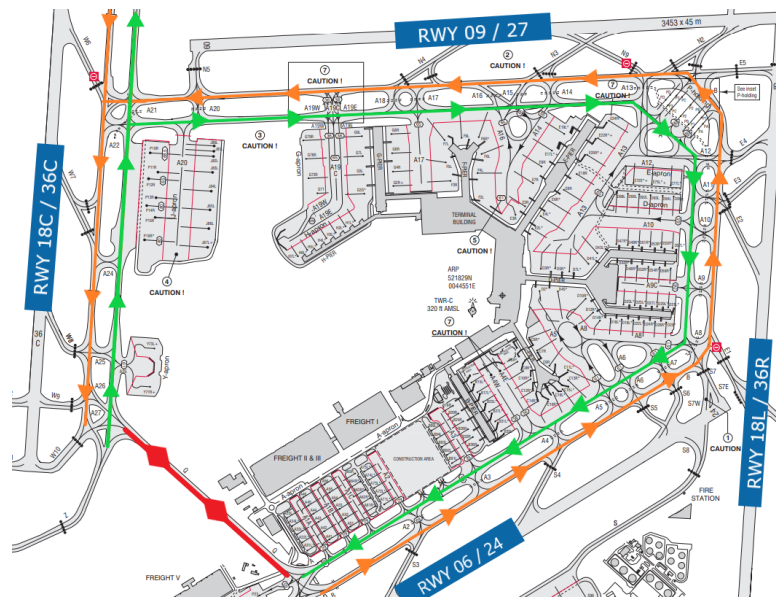


Figure 2.6: Taxiways Alpha (green, clockwise), Bravo (orange, counterclockwise) and Quebec (red). Adapted from LVNL [49].

As the taxiway is a complex network with many intersections, ATC uses standard methods in order to guide vehicles over the airport. One procedure that helps ATC with the traffic guidance is the standard taxi direction at the main taxiways around the passenger terminals, as shown in Figure 2.6. Taxiway Alpha is shown in green and is the inside taxiway in the clockwise direction. Taxiway Bravo is shown in orange and is the outside taxiway in the counterclockwise direction. These taxiways may only be used in the prescribed direction unless otherwise instructed by ATC [49]. Taxiway Quebec, shown in red in Figure 2.6, is a special taxiway and can only be used in one direction that can be adjusted by ATC. Schiphol is currently busy with the construction of an extra taxiway lane on taxiway Quebec, so that the entire ring around the passenger terminal is surrounded by dual taxiways [65].

While aircraft taxi over the airport, the flight crew is guided by ATC instructions via VHF radio communication. Unless otherwise instructed by ATC, pilots follow the standard taxi directions and adhere to a set of rules defined by the International Civil Aviation Organization (ICAO). These rules prescribe how to handle in case of potential collisions, close to runways and at stop bars and are listed below (taken directly from ICAO [39]):

- In case of danger of collision between two aircraft taxiing on the movement area of an aerodrome the following shall apply:
 - when two aircraft are approaching head on, or approximately so, each shall stop or where practicable alter its course to the right so as to keep well clear.
 - when two aircraft are on a converging course, the one which has the other on its right shall give way.
 - an aircraft which is being overtaken by another aircraft shall have the right-of-way and the overtaking aircraft shall keep well clear of the other aircraft.
- An aircraft taxiing on the manoeuvring area shall stop and hold at all runway-holding positions unless otherwise authorized by the aerodrome control tower.
- An aircraft taxiing on the manoeuvring area shall stop and hold at all lighted stop bars and may proceed further when the lights are switched off.

Although the standard procedures and ICAO rules make the routing of aircraft through the taxiway network convenient, they also limit the efficiency and capacity of the system. Aircraft could, for example, have to follow a longer route in order to adhere to the procedures than its theoretical shortest route. Systems that are not limited by cognitive capacity, could therefore be beneficial to make use of the infrastructure more efficiently.

2.2.3. Aprons and Gates

One of the key components of the airport's infrastructure are the aprons and the gates located on the aprons. As will become apparent in this section, many movements and events of the ground surface movement operation take place at or around the apron. Amsterdam Airport Schiphol (AAS) has 93 gates divided over 7 piers [68], as shown in Figure 2.7. An additional pier is currently being build with 8 extra gates. It will be placed on the left of the B-pier in Figure 2.7, further increasing the parking and handling capacity of the airport [74].

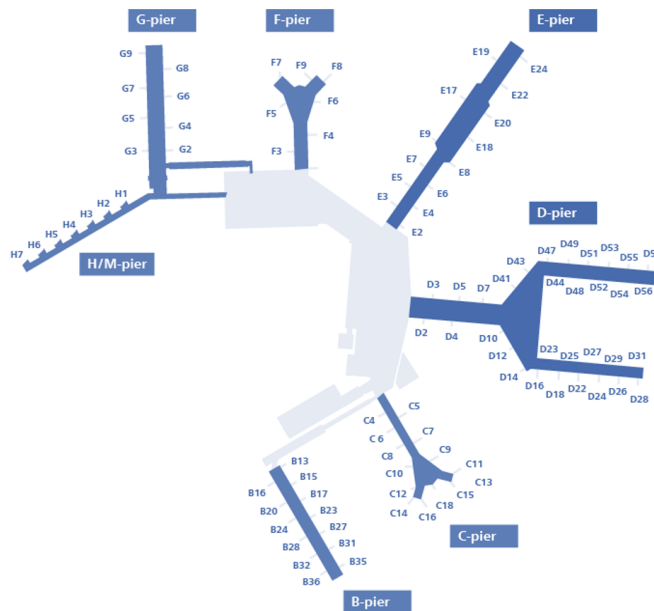


Figure 2.7: Overview of the existing piers at AAS [69].

In this section, the general infrastructure and usage of the apron and gates at Schiphol will be discussed. Besides that, the most important processes that take place at these locations such as the entrance and exit of the aircraft stands will be described. As this MSc thesis focuses on the operations at the apron specifically (and as a comparison will be made of a new concept with the conventional operation at the apron), it is imported to understand this in detail.

Gate Assignment

The usage of the gates is determined by the process of gate assignment, described in detail in the Regulation Aircraft Stand Allocation Schiphol (RASAS) [71]. Schiphol assigns the gates based on the different zones in the terminal (based on for example security or customs requirements) and by using the best-fit system. The best-fit system assigns the aircraft to the smallest possible stand available [71]. Additionally, connected gates, with a passenger bridge, always have the preference over bus gates. Although flights are allocated to a stand one day before their arrival, changes can occur just before arrival due to changes in the operation.

Once parked at the gate, the aircraft is allowed to occupy it for a maximum time before being reallocated to a parking position, as shown in Table 2.5. Most narrow bodies are part of the CAT 4 or lower category while most wide bodies are part of the CAT 5 or higher category. Additional conditions for considering gate allocation are that the minimum parking time at the remote location should be estimated to be 30 minutes or more. If these conditions apply and the gate capacity is needed for other flights, aircraft are obligated to be towed to a temporary parking position. This towing movement is then controlled by Apron Control from Schiphol airport.

Table 2.5: Maximum turn around time before gate reallocation is considered at AAS based on Schiphol guidelines [71].

Aircraft Category	Maximum Turn Around Time
CAT 4 or lower	170 minutes
CAT 5 or higher	210 minutes

Despite the reallocation obligation after maximum turn around times, it could happen that an inbound aircraft is assigned to a gate that is still occupied by another aircraft. This will be noticed by Air Traffic Control, who has the responsibility to direct incoming aircraft to their gates assigned by the airport. If this is the case, ATC usually redirects the incoming aircraft to a remote holding position where it waits. This process is referred to as inbound holding and will be discussed more elaborately in [subsection 2.2.4](#).

Aircraft Stand

Having explained the guidelines for the usage of the stand, this next subsection will focus on the aircraft stand itself and the entrance and exit of it. To make the best use of the airport its space, the aircraft stands are made so that an aircraft does not unnecessarily block any valuable space when parked. This means that many different actors are present in a small area, from baggage handlers, caterers, refuelling services to push-back trucks. To accurately model the apron operations at an airport, it is important to understand the restrictions on the entry and exit of the aircraft stands. This subsection will briefly discuss these.

An overview of a typical stand at Schiphol and the signage is shown in [Figure 2.8](#). The signage on the ground of the apron area and of the stands constrain the movements and warn actors for hazards present in this small area, such as indicating an area that has to be kept free.

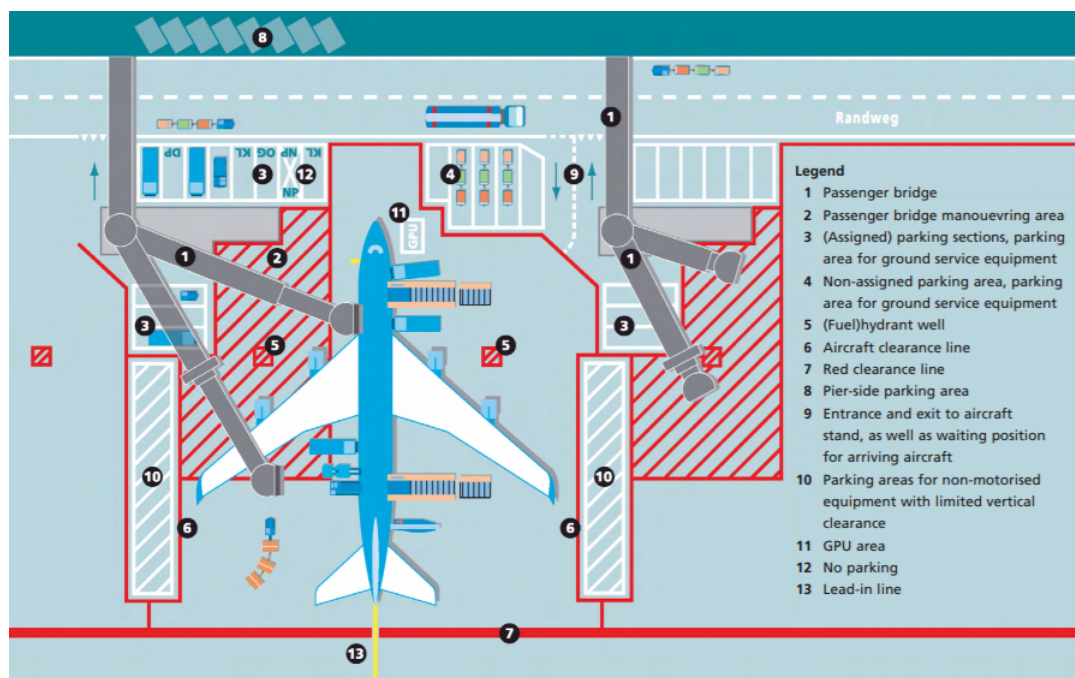


Figure 2.8: Overview of the aircraft stand and signage used [66].

Number 13 shows the lead-in line, indicating the path the aircraft has to take when it enters the stand so that it does not hit any obstacles. All ground handling vehicles have to enter the stand at number 9, except the push-back truck or the marshaller, who are allowed to enter the stand in front of the aircraft's nose [66]. Number 6 shows the aircraft clearance line, ensuring that aircraft have enough space to enter and leave their stand. Number 7 shows the red clearance line, which indicates the boundary between the ATC controlled and the aircraft its stand. [Figure 2.9](#) shows this red clearance line from another perspective, where it can be seen that it clearly separates the stands from the ATC controlled area, connected to the taxiway system, outside of these lines.

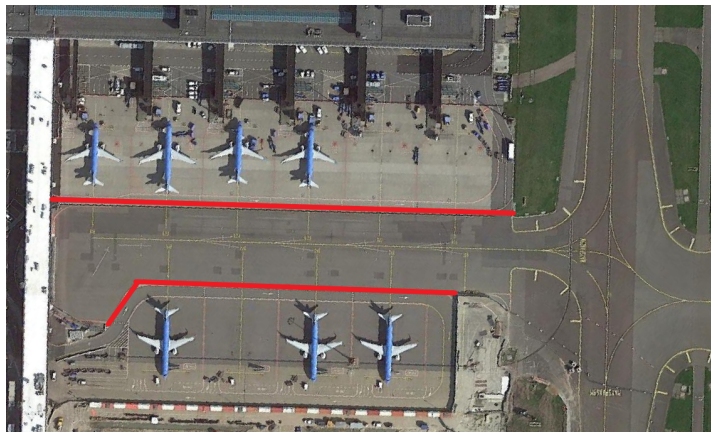


Figure 2.9: Satellite image of the west-side of the B-pier showing the ATC boundary by the red line. Adapted from Google Earth [20].

This boundary forms an important interface in the system where both the taxiway system and the apron with the aircraft stands are considered. As both areas are controlled by different actors, permission is required to move over the red boundary line. The crossing from both directions will be described in detail below:

The crossing in the direction from the taxiway system towards the aircraft stand can only be made when permission is indicated by a marshaller or by the visual docking guidance system [49] (VDGS). This VDGS is an electronic system used at most gates at Schiphol to give instructions to pilots when they enter or leave the stand.

As described in the Aeronautical Information Package (AIP), [49], the pilot may only enter the stand if the VDGS display is illuminated and when the right aircraft type is shown on this display. Sensors in the apron will then sense the position of the aircraft relative to the stand and give feedback to the pilot whether to stay on the line, move to the left or to move to the right and indicate this on the VDGS display. The system will also give an indication to the pilot how far the aircraft still has to move forward and it will finally indicate when to stop. In its simple version, the visual guidance docking system has to be activated by the ground handler. Automated activation of the visual guidance docking system at the gate is part of the guidance service of advanced surface movement guidance and control systems (A-SMGCS) that will be discussed later in this report.

The crossing of the red boundary line in the direction out of the stand, can only be made when a clearance is obtained from ATC. This is usually combined with the push-back procedure which will be described in the next subsection.

Push-back and Engine start up

Most aircraft that want to leave their gate at Schiphol are required to do so by using a push-back truck. This is the case as at the majority of aircraft stands, the aircraft is positioned with front of the aircraft pointing to the gate, also called nose-in parking. When an aircraft leaves a nose-in gate, a push-back manoeuvre is used where the aircraft is pushed backwards by a push-back tug. An exception is the B-pier, where there is the possibility for aircraft to leave the gate autonomously. The layout of these gates enables the so-called "taxi-in-taxi-out" [70]. The push-back procedure for the nose-in gates will be described below.

The push-back procedure is a highly standardized process both regarding the procedures that are followed and the paths that have to be used. The standard push-back procedure to be followed during regular visibility conditions is summarized in Figure 2.10 based on the definition by Schiphol [72].

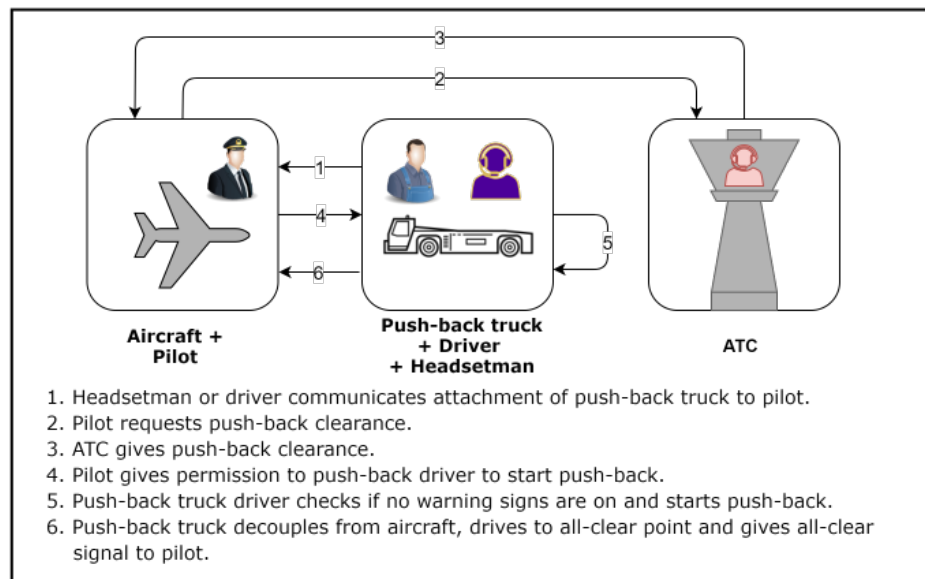


Figure 2.10: Standard push-back procedure at AAS based on Schiphol procedures [72].

The push-back procedure for the J-apron is always different from the standard procedure as this platform is not under control of LVNL. In this case, the push-back clearance is given by Apron Control who orders the aircraft to contact LVNL Ground Control after the push-back is finished. The standard push-back procedure can be adapted for all aprons in case of low visibility conditions at the airport or in case of de-icing, but as these operations are out of the scope of this study this alternative procedure will not be discussed.

The push-back path per gate is also standardized and an overview of the paths per gate and aircraft category is publicly accessible [72]. The map specifies the path of the push-back and any particularities such as conflicts with active taxiways during a push-back manoeuvre. More detailed maps are available on a gate level similar to the examples as given in 2.11(a) and 2.11(b) [75]. The push-back ends with the aircraft parked with its nose wheel on the push-back limit line, as indicated by the red rectangle with white circle in 2.11(a) and 2.11(b). With the aircraft in the correct position, the driver of the push-back truck decouples and parks the push-back truck at some distance from the aircraft where he gives an all-clear sign to the pilot indicating that the aircraft is free to commence its taxi procedure.

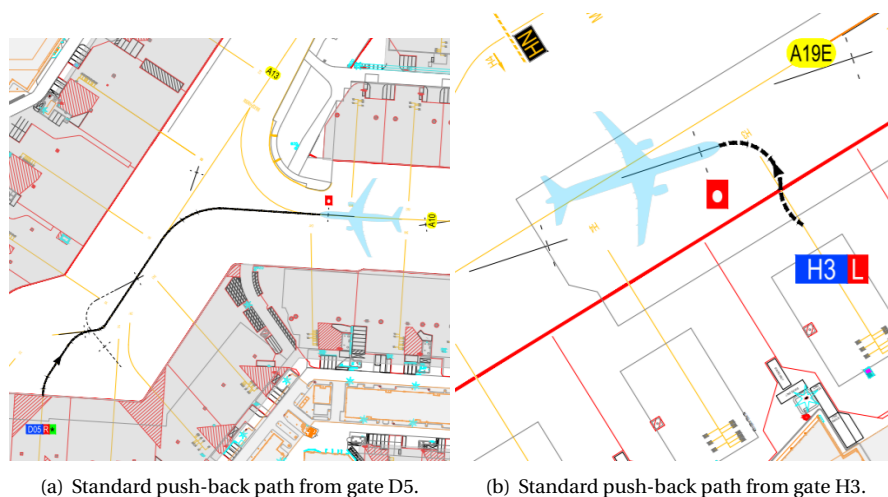


Figure 2.11: Standard push-back paths from two example gates at AAS [72].

In the conventional operation, the engine start up usually takes place during or right after push-back. Push-

back paths are specified in such a way that the engine blast created during engine start does not damage any buildings or vehicles parked at other aircraft stands. For some gates however, a push-pull manoeuvre is required where the aircraft is first pushed back and then pulled forward by the push-back truck in order to place it in a location where the engine blast will not cause any damage to obstacles in the surrounding.

If taxiing from the runway to the gate on its own power, the aircraft needs to have its engines started and stabilized to leave the gate. In the conventional operation, this is usually done during or right after push-back. Engine start during push-back is only possible when the engine start up time is less than the duration of the push-back. If not, the aircraft begins the engine start up during push-back and finishes it while standing still at the end of its push-back path. For aircraft leaving a gate that requires a push-pull manoeuvre, engines can only be started after final standstill which means that the aircraft will block a part of the apron for a duration equal to its engine start up time. The duration of the push-back/push-pull and the time needed for engine start determine the time the aircraft needs before it can start taxiing. The three different scenarios are listed in Table 2.6, dependent on whether push-back or push-pull is used and on the relation of the engine start up time to the time needed to move the aircraft over the push-back or push-pull path.

Table 2.6: Three scenarios for the time before an aircraft can start taxiing after push-back and engine start.

Push-back or Push-pull	Engine start up time (ESUT)	Time required before aircraft taxiing can start
Push-back	$ESUT \leq \text{push-back time}$	Push-back time
Push-back	$ESUT > \text{push-back time}$	ESUT
Push-pull	ESUT	Push-pull time + ESUT

Data about the average duration of the push-back per gate was requested from operational experts at Amsterdam Airport Schiphol but was not available. It was therefore determined that the duration of the push-back can be determined based on the detailed push-back paths as shown in Figure 2.11 (on scale) and the average speed during push-back. The maximum speed for conventional push-back is restricted by the forces exceeded on the nose landing gear of the aircraft and is reported to be 3 kts [101]. The engine start up time differs per aircraft type and is dependent on the size and number of engines.

2.2.4. Remote holding locations

AAS also has some dedicated locations that are indicated as remote holding locations. These locations fulfill a special task in case an assigned gate for an incoming aircraft is still occupied by another aircraft. In this case, LVNL can decide to redirect the incoming aircraft to one of the holding positions as shown in Figure 2.12, except P4 or P5 which are not used. This process is referred to as inbound holding.

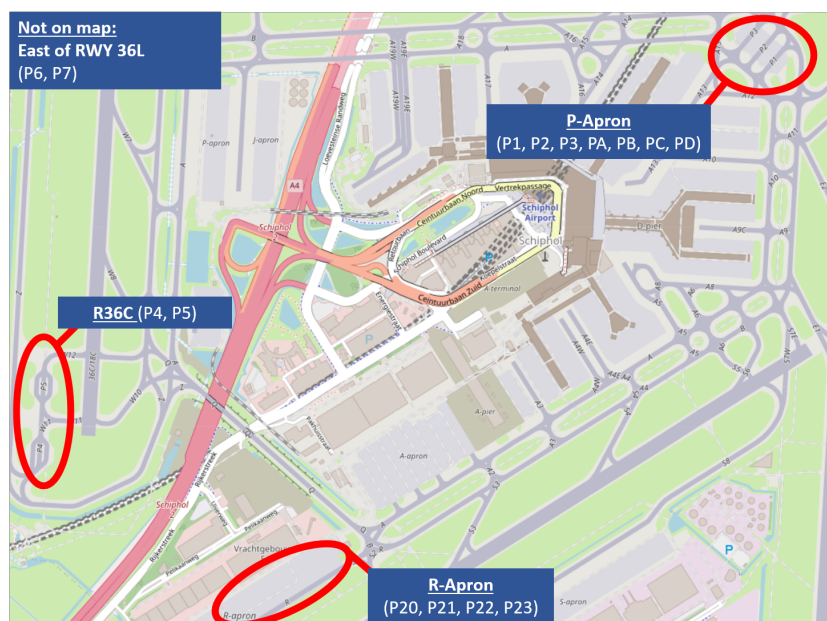


Figure 2.12: Holding positions at Amsterdam Airport Schiphol. Adapted based on map from OpenStreetMap contributors [57].

The usage of the positions is restricted by the maximum wingspan which is shown in Table 2.7. An additional restriction is that either the combination of PA and PB can be used or P1. The same applies for the combination of PC and PD or P3.

Table 2.7: Maximum wingspan restriction per position (N.A. indicating no restriction) [49].

Position	P1	P2	P3	PA,PB,PC,PD	P4	P5	P6	P7	P20	P21	P22	P23
Max wingspan [meters]	69	36	N.A.	36	N.A.	N.A.	N.A.	N.A.	36	36	65	65

According to the aeronautical information package for Schiphol, another use case for the remote holding positions is when a departing aircraft has to vacate a gate for an incoming aircraft while its departure is constrained by a CTOT from network control [49]. Interviews with experts from AAS however made clear that this is a very rare solution in the current operation. They indicated that inbound holding of the arriving aircraft is more common in the situation where a departing aircraft is still occupying the gate. This leads to a so-called "double holding delay" where an inbound aircraft is waiting for an outbound aircraft that is waiting for permission to leave the gate. This is an undesirable situation which is the reason for the consideration of a new process of outbound holding. This will be discussed in more detail in chapter 3.

2.2.5. Service roads

The last key element of the airport that will be discussed are the service roads. Although the aircraft use the taxiway network to move over the airport, there are many other vehicles part of the ground surface operations. Parties such as ground handlers, bird controllers or emergency services do also move around the airport. These parties mainly use the service roads. Schiphol has defined special rules for vehicles on the airside of the airport, listed below and taken from the Schipholrules document [80]:

- Vehicles are not allowed to enter aircraft taxiways, unless they use the predefined crossings.
- The maximum speed on service roads and platforms is 30 km/h.
- Vehicles on the service roads around the piers have right of way on vehicles from the platforms.
- Vehicles on taxiways have right of way on vehicles crossing the taxiways.
- Vehicles in the landing area or on the platform have a predefined priority where vehicles lower in the list should give right of way to vehicles higher in the list:
 1. Aircraft that are taking off or landing.
 2. Emergency vehicles with siren and lights on.
 3. Taxiing aircraft or hovering helicopters including vehicles guiding them.
 4. Passengers who are guided to or from an aircraft.
 5. Towed aircraft.
 6. All other vehicles.

Note the fairly low priority of towed aircraft while moving over the airport. It should therefore be considered how to handle priority of aircraft that are being towed by a TaxiBot, which will be modelled in this study.

2.2.6. Concluding Remarks

In the previous section, the airport ground surface operation at Schiphol has been discussed. First the five main runways at Schiphol were considered with a description of their preferred usage sequence and an analysis of the usage of each of the runways. Then the taxiway network was described that forms the connection between the runways and the gates. The standard procedures were discussed and it was argued that the taxiway system can potentially be used more efficiently if coordinating entities are not limited by cognitive capacity. Also the aprons and the gates were discussed as an important element of the airport ground surface operation. The rules for the assignment and the occupancy of the gates are fixed and are used to determine where aircraft can be parked or when aircraft should be relocated from their gate. It was also shown that the movement from the stand onto the apron is standardized to a high degree by using standardized push-back paths and that this process is strictly coordinated by ATC to prevent collisions. Remote holding locations were discussed as a place to facilitate holding of aircraft that are assigned to an occupied gate and the problem of a so-called double holding delay was raised. Finally, the use of the service roads at the airport was discussed and it was noted that the current rules would have to be adapted for effective routing of TaxiBot-towed aircraft over the airport.

2.3. Traffic Management Techniques

The next section will elaborate on traffic management techniques, that are used to prevent bottlenecks in the airport's taxiway network. At large airport such as Amsterdam Airport Schiphol, bottlenecks can occur during busy periods. Examples are queues that form in front of the runway and inbound aircraft that have to wait for their gate because it is still occupied by another aircraft. This is undesirable from an efficiency perspective, as taxi times increase, but also from a sustainability perspective as the fuel burn increases due to the increased engine on time. Several techniques are therefore used at airports in order to prevent such bottlenecks and will be discussed in this section. As will become apparent when the techniques are discussed, they all influence the operations on and around the apron significantly.

2.3.1. Collaborative Pre-Departure Sequence Planning (CPDSP)

The introduction to A-CDM already briefly covered the collaborative pre-departure sequencing element of A-CDM that is used at AAS. An overview of the system is shown in Figure 2.13. The use of the CPDSP tool can result in a startup delay as part of the optimal departure sequencing plan. This is shown in blue in Figure 2.13 and is also referred to as gate holding. The result of introducing a start up delay and thus holding it at the gate could influence arriving flights as will be discussed in the next section. In the CPDSP system, the target start-up approval time (TSAT) is calculated based on the target off-block time (TOBT). Flights participating in the CPDSP system should comply with certain conditions as specified in the Schiphol A-CDM manual [77]. The most relevant three are that the flight must be an IFR flight, a TOBT must be available and the flight must be planned to depart from one of the main runways (all runways excluding RWY 04/22). The CPDSP system first calculates an optimal sequence in which aircraft are scheduled for take-off, initially based on the sum of the TOBT and the estimated taxi time (EXOT). Besides that, it takes into account the CTOT if issued by EUROCONTROL, the Standard Instrument Departure route (SID), Runway capacity and the required separation to prevent wake turbulence. Based on this information, a sequence of departing aircraft with a target take-off time (TTOT) can be issued. Then subtracting the estimated taxi out time (EXOT) from this, gives the TSAT for each specific aircraft.

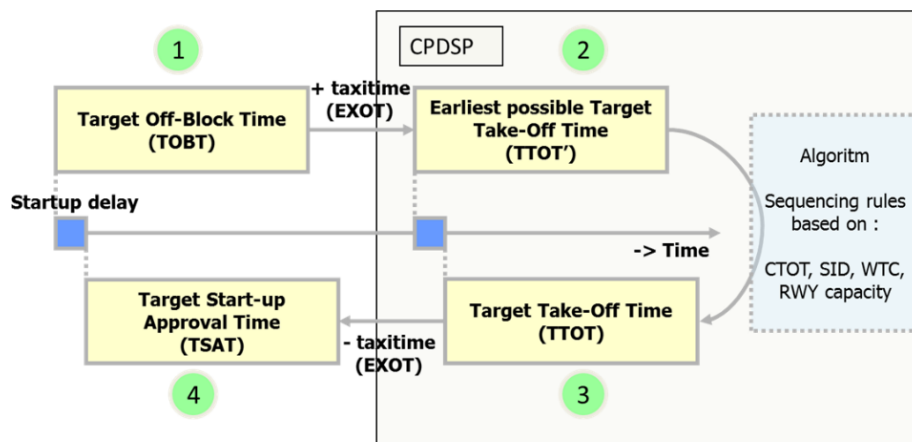


Figure 2.13: Overview of the Collaborative Pre-Departure Sequence Planning system at AAS [77].

The startup delay that can be introduced as a result of this algorithm is a technique that is more generally known as gate holding or stand holding. This technique will be discussed in the next section.

2.3.2. Gate holding

As discussed above, Amsterdam Airport Schiphol uses the collaborative pre-departure sequence planning (CPDSP) in order to determine the start-up approval time, the time an outbound aircraft leaves the gate. Holding at the gate is a result of this planning. Gate holding is discussed extensively in literature. For example, Ravizza et al. [60] developed an algorithm that lets aircraft wait at the gate if necessary instead of clearing them to taxi and wait on the taxiways with their engines on. They used Zürich airport as a case study. This was shown to lead to less fuel burn than in the actual operation. A study into gate holding by Hao et al. showed similar results [32]. However, there are also disadvantages of gate holding. Kim et al. [44] modelled

scheduled arrivals and departures for New York La Guardia Airport (LGA). Gate holding was tested with the gate assignment obtained from the real operational data and was found to reduce the taxi-out times while a shift to increased gate delays was observed. It was also shown that gate holding causes gate conflicts, where inbound aircraft are assigned to a gate that is being occupied by an outbound aircraft hold at the gate. Although detailed studies about the effect of gate holding on the number of gate conflicts and gate delays are not available, experts from Amsterdam Airport Schiphol indicate that gate holding has similar effects at this airport especially during hours with a lot of traffic. For this reason, the system wide effects of holding an outbound aircraft at the gate should be considered well before deciding to do this.

Another example of holding aircraft at the gate is push-back rate control [89]. In this case, the startup time at the gate is not a calculated value but rather a result of a limited number of push-backs that is allowed in a certain period of time. This technique was shown to distribute the peak demand for push-backs and resulted in lower taxi-out times [89]. The authors note that push-backs were allowed if an incoming aircraft needed the gate but found out that this was rarely the case on Boston Airport where they based their case study on.

2.4. Concluding Remarks

Conventional ground surface operations were discussed in this first chapter. It was shown that many stakeholders are involved in the planning and coordination of the ground surface operation. Timing milestones from the Airport Collaborative Decision Making (A-CDM) milestone approach are used by these stakeholders to share situational awareness and predict future events. Amsterdam Airport Schiphol was found to be a suitable airport to base a case study on for this research, due to its relatively large size and high traffic numbers. It was shown that planning and coordination of the conventional ground surface movement operation is a task that is demanding for Air Traffic Control Officers (ATCO's), which is why they rely on standard procedures. Standard taxi directions are for example used in the taxiway system. With an expected increase in air traffic, the task of efficient and safe planning becomes more challenging. Developments in ground surface operations are taking place and will be discussed in the next chapter. It will become clear that, while some developments assist ATCO's in their tasks, other developments make the task of planning and coordinating traffic over the airport more challenging.

3

Developments in Ground Surface Operations

Up to this point, the current ground surface operations have been discussed with a special focus on Amsterdam Airport Schiphol. Although the current systems do provide a safe ground surface movement operation at airports, both efficiency and safety can still be improved. Furthermore, the role of sustainability in the aviation industry is increasing. Autonomous and electric systems could play an important role in achieving a safer, more efficient and more environment friendly ground surface operation. The Single European Sky ATM Research (SESAR) joint undertaking [81] aims to contribute to this. Besides that, the Royal Schiphol Group has developed a vision for 2050 with sustainability goals involving more autonomous operations [76]. These projects will be introduced briefly below.

The SESAR joint undertaking involves many projects that contribute to a smart and sustainable aviation system for Europe. Two of the projects part of SESAR that are of relevance for this research are the AEON project and the improvement of the advanced surface movement guidance and control system (A-SMGCS) at Schiphol. The focus of the Advanced Engine Off Navigation (AEON) project is on the operational concept of more environmentally friendly taxiing techniques such as TaxiBot, E-Taxi and single engine taxiing [2]. This research is done in parallel with the start phase of the AEON project, enabling a collaboration with the stakeholders in the project. The other SESAR project that will be discussed in this section is the improved advanced surface movement guidance and control system (A-SMGCS) at Schiphol, which forms an example of a shift from centralized control to more distributed control.

Next to the SESAR projects, developments do also take place at a more local level. The Royal Schiphol Group aims to operate the world's most sustainable airports by 2050 as published in the vision document "Sustaining your world" [63]. One of the goals is to operate emission free ground surface operations at airports by 2030, while at the same time increasing the capacity of the airport and keeping a safe operation ensured. Another program which was launched by the Royal Schiphol Group is called Autonomous Airside Operations [76], in which they envision a fleet of autonomous electric vehicles that can interact with each other to fulfill the operations at the apron. Trials with sustainable taxiing by means of electric towing vehicles called TaxiBots are part of this development and will be discussed further in this section.

3.1. Advanced Surface Movement Guidance and Control Systems (A-SMGCS)

An Advanced Surface Movement Guidance and Control System (A-SMGCS) is a system used to enable ground surface operations at airports under all weather conditions. As part of the SESAR project, Air Traffic Control the Netherlands (LVNL) is currently working on the improvement of the existing A-SMGCS system at Amsterdam Airport Schiphol where the surveillance system is being updated [82]. The core elements of A-SMGCS are described in manuals by both ICAO and EUROCONTROL ([41], [24]) and an overview of the system is visualized in Figure 3.1. The system architecture is discussed in this report as it is a state-of-the-art example of a shift from centralized control to more distributed control with ATCO's in a supervisory role. As shown in Figure 3.1, the A-SMGCS services can be split into four different services: the surveillance service, routing ser-

vice, guidance service and airport safety support service. This subdivision is the one used by EUROCONTROL [24] and will be used for this report. After introducing the required data sources for the A-SMGCS system, a more detailed descriptions of the different services is given below

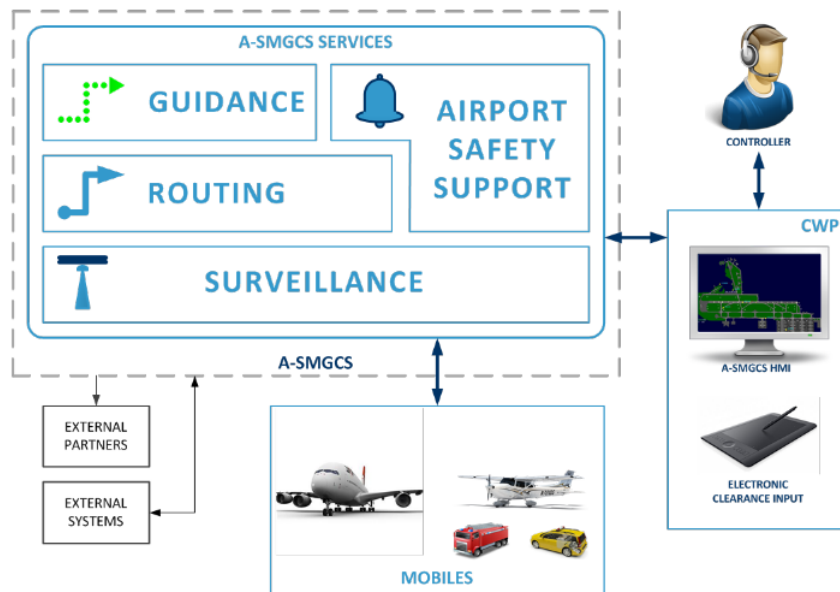


Figure 3.1: Overview of the 4 services of the A-SMGCS system as specified by EUROCONTROL [24].

3.1.1. Data required for the A-SMGCS system

An important requirement for the successful implementation of an A-SMGCS system is the availability of data. There are two major data sources for the system to work, which are the mobile information database and the airport operation status database. The mobile information database contains information about the aircraft or vehicle such as the identity, the type of aircraft or vehicle, planned route and received ATC clearances. The airport operation status database contains information about the airport infrastructure such as runways and taxiways or information about common procedures at a certain airport. Besides that, information about weather and anticipated changes in runway or taxiway configurations is shared in this database.

3.1.2. Surveillance Service

The A-SMGCS surveillance service provides information about the location and movements of all vehicles and obstacles on the airport. Data about the airport infrastructure such as a map including its obstacles is required from the airport operation status database for this service. Flight data such as identity and clearances received and vehicle information such as the planned route is required from the mobile information database for this service. A distinction is made between cooperative and non-cooperative vehicles and obstacles. Information about cooperative vehicles, including their identity is shared by the vehicle with the A-SMGCS system automatically at all times. An example of a cooperative vehicle is an aircraft with an active transponder. Information about non-cooperative vehicles is not shared with the A-SMGCS system automatically, which means its identity is not automatically known. An example of a non-cooperative vehicle is an aircraft with a broken or switched off transponder. A requirement for the surveillance service is that both cooperative and non-cooperative vehicles must be detected by the A-SMGCS system. This means that the airport should install sensors in order to collect data about the location and movements of non-cooperative vehicles not sharing this information automatically. Examples of sensors to collect data about non-cooperative vehicles are surface movement radars, optical sensors or magnetic sensors [24]. The main function of the surveillance service is to combine the data from non-cooperative and cooperative sensor systems.

The surveillance service enables controllers to be aware of the situation at the airport regardless of the visibility conditions. The other three services will be discussed next and require the implementation of the surveillance service as a precondition.

3.1.3. Airport Safety Support Service

The airport safety support service is a service that can send out alerts to the users of the A-SMGCS system. It combines the data from the surveillance service with data from the mobile information database and the airport operations status database. Two aircraft might for example be cleared to land on the same runway at the same moment in time. The system will detect this and alert the controller so that he or she can correct this situation. The main functionalities of the support service include monitoring of runways and conformance with ATC instructions and alerting services. Alerting services will notify ATCO's in case a conflict is anticipated, or when vehicles deviate from their cleared paths.

3.1.4. Routing Service

The routing service provides routes for aircraft or ground vehicles on an airport while taking into account the clearances by ATC and potential constraints such as the status (active/non-active) of runways and taxiways. It is an example of a system that reduces ATC workload by shifting tasks from a more controlling nature to supervisory nature. The most basic information that the system requires is the start and endpoint of a vehicle's route, which is usually defined by either an aircraft stand or a specific runway entry/exit for aircraft. Besides that, flight characteristics such as the aircraft type, identification estimated arrival time should be shared. The system also requires information from the airport operation status database about the layout of the airport, standard procedures or potential constraints for taxi routes. An important condition for the implementation of the routing service is that the surveillance service is already implemented for the constant supply of the position and identification of vehicles on the airport.

Next to the generation of an aircraft route, the routing service can also be used to create an alternative route for an aircraft that has already received route clearance. This could happen in case a taxi lane on the planned route that was available becomes unavailable due to, for example, an newly opened runway.

By using the routing service, taxi times can also be predicted. This information can then be shared with the A-CDM system as described in earlier in this report in order to calculate the estimated take off time for flights that take-off or the estimated in-block time for arriving aircraft.

3.1.5. Guidance Service

The guidance service is used to guide aircraft or other ground vehicles visually. It is an example of a means that shifts central control to local control. Taxiway center line lights (TCL) can for example be switched on and off automatically to visualize the route that an aircraft or vehicle has to follow. Also stop bars can be switched on and off automatically, based on clearances given by the ATCO in the HMI. The previously discussed VDGS system, used to indicate clearance to enter the aircraft or to help to pilot to align its aircraft on the stand can also be switched automatically with this guidance service.

Especially the routing and guidance service as part of the A-SMGCS system form a basis for shifting control to a more local level. This could assist ATCO's and therefore reduce their workload. Although the above described architecture is described well by EUROCONTROL, the techniques that make the routing and guidance service possible have not yet received many attention. Previous research that did consider this decentralized concept of controlling airport ground service movement operations will be discussed in [section 4.3](#).

3.2. Engine-Off Taxiing

One of key elements to achieve a more sustainable ground surface operation is reducing the engine on time of aircraft. The advanced engine off navigation (AEON) project that was previously discussed aims to investigate solutions for more environmentally friendly taxiing by looking at solutions such as TaxiBot, E-Taxi and single engine taxiing [2]. On a more general level, a distinction is made between internal and external electric taxiing systems, where internal electric taxi system are part of the aircraft and external taxiing systems couple and uncouple to the aircraft. The advantages and disadvantages of these systems will be discussed briefly in the first subsection. One engine-off taxiing concepts will be chosen at the end of the first subsection to be considered in more detail in this study. From that point onward, the operational concept, technical specifications and differences with the conventional operation will be elaborated. Finally, the challenges in the implementation of this new concept at Amsterdam Airport Schiphol will be outlined.

3.2.1. Engine-Off Taxiing Techniques

B. Benda [9] did an extensive analysis of the state-of-the-art engine-off taxiing systems reviewing both several internal systems and external systems. The feasibility of internal taxiing system is questioned for large airports in this work. To backup this claim Benda refers to Okuniek et al. [56] who state that the extra weight of internal systems will not be beneficial for long-haul flights and to Hospodka [37] who states that an internal system increases the maintenance costs of the aircraft. Although there are internal systems that have been tested, none of the systems is certified to be used in the operation yet.

External systems on the other hand, do not add any additional weight to the aircraft. Two of such systems were reviewed by B. Benda, one using an external tow-bar and another using a new vehicle called TaxiBot. In both of these options the aircraft would be towed to a location near the runway instead of starting its engines at the gates. Benda references Guo et al. [31] who describes the increased fatigue loads on the nose landing gear that shorten the life cycle with towing using an external tow-bar. This is found to be a major disadvantage of a tow-bar system. The final system considered by Benda is the Taxibot, lifting the aircraft's nose wheel from the ground while towing. This system has a main advantage over the tow-bar system due to its design where the nose landing gear limits are not exceeded during towing [9]. Benda references Guo et al. [31] to identify a major disadvantage of the TaxiBot as a engine-off taxiing concept, which is the increased operational complexity of the concept. Moreover, Benda finds that TaxiBot is the most promising concept to be considered for implementation at Amsterdam Airport Schiphol mainly due to the fact that it is the only certified system [9]. Other research also identifies TaxiBot as one of the most promising engine-off taxiing concepts (e.g. van Baaren and Roling [98] and Soomers [90]). Finally, a recent case study performed at Amsterdam Airport Schiphol concluded that the TaxiBot concept is infeasible in its current form but might be feasible when preconditions are met [79]. These preconditions contribute to the interest to explore this concept in more detail in this study.

For the scope of this research, it is chosen to consider engine-off taxiing with TaxiBots. It is currently considered to be the most promising concept based to be implemented on the short term based on the research discussed above. Furthermore, small-scale case studies with this concept at Schiphol have identified preconditions that need to be met before the concept can be implemented. This adds to the relevance to explore the concept on a larger scale in this study. The remainder of this chapter and research will therefore focus on the TaxiBot concept in more detail.

3.2.2. TaxiBot Concept and Technical Specifications

The TaxiBot is an external vehicle that can be attached to the aircraft for push-back and towing the aircraft and it looks similar to towbarless push-back trucks that are currently being used at Schiphol. There are two types of TaxiBots, the narrow body (NB) TaxiBot, suitable for Airbus A318 to A321 and Boeing B737 to B757 aircraft and the wide body (WB) TaxiBot which is suitable for Airbus A330 to A380 aircraft and Boeing B767 to B747 aircraft. The only certified version of the TaxiBot, as shown in Figure 3.2, is currently the NB for the Boeing 737 (classic and next generation) and Airbus A320 family.



Figure 3.2: TaxiBot towing an aircraft at Schiphol Airport [45].

Where conventional push-back trucks can only be used for the push-back of aircraft and towing of empty aircraft, the functionality of TaxiBot is more extensive than that. The main technical differences are described below based on the concept owner's website [38]. After push-back by the TaxiBot and its driver, the control can be shifted from the driver to the pilot in the aircraft who can then use the TaxiBot to taxi to the runway instead of starting its engines and using those to taxi. The pilot uses the steering wheel and the brakes in the cockpit in a similar way as with conventional taxiing, while still achieving maximum taxi speeds of 23 kts, that are similar to the conventional taxi speeds at many airports. Conventional push-back trucks do not have this pilot-control-mode which raises problems from a safety and accountability perspective when used for taxiing. Another problem with the use of conventional push-back trucks for taxi towing is the life time of the nose landing gear that is reduced. TaxiBot does not have this problem as it is claimed to have no impact on the fatigue life of the nose landing gear. An additional benefit is that the chance of foreign object damage in the engines is claimed to be lower with TaxiBot taxiing when compared to conventional taxiing.

Now that the technical specifications of the TaxiBot have been discussed together with its implications for individual aircraft. The implementation of TaxiBot at a large airport such as Amsterdam Airport Schiphol will also have operational effects. The main differences with the conventional operation and the challenges in the implementation of TaxiBot will be discussed next.

3.2.3. Main Differences TaxiBot Taxiing Operations and Conventional Taxi Operations

Now that the individual TaxiBot vehicle has been discussed, the airport ground surface movement operation with TaxiBots will be compared to conventional operations. The main differences for an outbound flight can be grouped in three different categories, as shown below. In [subsection 3.2.4](#), the challenges associated with the implementation of TaxiBot will be discussed in more detail.

- **Push-back, Taxi-out and Engine Start:** In the current operation, push-back is done by an external push-back truck with a driver. This truck is uncoupled right after push-back when the aircraft is in the taxiway network and has started its engines. With TaxiBot, aircraft are pushed back by a TaxiBot with a driver. Although the majority of the push-back procedure is the same, a significant difference is that the TaxiBot does not uncouple right after push-back and that the aircraft's engines are not started yet. When a TaxiBot is used, the aircraft taxis to a uncoupling location close to the runway powered by the TaxiBot. During taxiing or at the uncoupling location, the aircraft starts its engines. TaxiBot is claimed to increase the efficiency at the airport due to the improved throughput at airport apron not having to start their engines [38], though this has never been tested on a large airport with many TaxiBots and aircraft. However, uncoupling the TaxiBot from the aircraft has to be done near the runway instead. This could move the congestion from the apron towards the uncoupling location that would not be a problem in case of conventional taxi operations.
- **Increased Traffic Density:** The result of this uncoupling at the runway is that TaxiBots have to be routed back from the uncoupling location back to their parking position or new task. The traffic density thus increases as compared to the conventional taxiing operation and therefore increases the workload of air traffic controllers responsible for this.
- **Environmental and Economic Impact:** As described before, the use of TaxiBot could decrease the time that the aircraft has its engines on and therefore decrease the fuel used. While TaxiBot reseller Smart Airport Systems claims that fuel consumption can be reduced up to 85% as compared to the conventional operation [96], operational tests at Schiphol have shown an average reduction of around 50% [79]. It should be noted that this trial at Schiphol was only based on 7 trials in which one of the closest runways (RWY 24) ways over-represented. It is expected that gains could be higher for runways with longer taxi times such as for example runway 36L.

These differences lead to several new challenges in the operation, as will be elaborated in the next subsection.

3.2.4. Challenges in the Implementation of TaxiBot

To assess the feasibility of the implementation of the TaxiBot at AAS, several studies have been conducted from an industry perspective already. First of all, operational trials with a single TaxiBot were performed at Schiphol in collaboration with different stakeholders. The results are summarized in the TaxiBot proof of concept [78]. Besides that, Schiphol asked aviation consultant TO70 to do a simulation study on the implementation of TaxiBots in the current operational context [28]. For this simulation, it was assumed that the

concept is introduced in the current operational context. Schiphol bundled these two studies into one feasibility study [79] with the aim to find out whether sustainable taxiing with TaxiBots is feasible at Schiphol in 2030. The concept studied was a full-scale TaxiBot implementation, meaning that both all inbound and all outbound flights were assumed to be towed by TaxiBots. The conclusion of the report was that the concept would be infeasible in its current form, but that it could be feasible in the future with adjustments [79]. The main precondition that was identified is related to the control of traffic at the airport. As already discussed in the previous subsection, the implementation of TaxiBots increases the traffic density at the airport due to TaxiBots having to move back from the decoupling points close to the runways. Besides that, the traffic flow at the airport will change because of the uncoupling of aircraft at the runway, thereby blocking these locations for other traffic temporarily. ATC indicated that, based on the results from the simulation study and experiments, they would not be able to maintain current capacity, safety, on time performance and acceptable workloads when the proposed concept would be introduced in the current system [79]. The feasibility study by Schiphol suggests steps in the direction of autonomy to tackle these issues. Benda [9] has explored a first concept in this direction.

Benda [9] studied the concept of autonomous towing at Amsterdam Airport Schiphol based on distributed coordination and planning in an academic study. He performed a simulation study in which all outbound aircraft were assumed to taxi with TaxiBots. Benda showed that distributed coordination and planning is a successful technique to control and guide vehicles over the airport when TaxiBots are used for outbound taxiing. More information on this study is reported in [chapter 4](#) in a comparison between previous studies that use a similar planning approach. Although Benda's study provided first insights in the effects of the implementation of TaxiBot at AAS, a simplified airport layout was used. Gates and aprons were simplified in the model. The result of this simplification was that TaxiBot operations at the apron, such as push-back, could not be simulated. Benda therefore recommended to include aprons and gates so that the TaxiBot operation can be compared to conventional operations more accurately. The lack of gates and aprons in the model used by Benda did not allow to simulate runway scheduling and to incorporate holding at the gate. In [section 2.3](#), gate holding was identified as a frequently used approach to manage traffic at the airport in general. With the inclusion of TaxiBots and the resulting increased traffic density, it is therefore interesting to evaluate the option of holding and the results thereof in this study.

The effects of full scale TaxiBot implementation at an airport such as Amsterdam Airport Schiphol have thus not been researched extensively. Case studies and simulation studies performed by or for Schiphol have identified challenges in maintaining capacity, safety, on time performance and ATC workload as a result of TaxiBot implementation in the current system. At the same time, autonomy could help to increase the capacity and efficiency of the airport and therefore offers interesting new opportunities for both a more efficient and a more sustainable airport ground surface operation. Exploratory research by Benda [9] has shown that distributed planning and coordination can be used successfully to plan and control vehicles at an airport while TaxiBot is used for outbound taxiing. Apron operations were however excluded from the study. The addition of these operations would allow for a more accurate comparison of TaxiBot with the conventional operation.

3.3. Novel Traffic Management Techniques

As discussed previously in this report, there are several ways to manage the flow of traffic over the airport. New taxiing techniques such as TaxiBot enable new techniques that might be more efficient and lead to less fuel burn. Also, the development of the improved A-SMGCS system might allow for the introduction of new concepts, as the set of possibilities to monitor and guide aircraft is extended and more accurate. Novel traffic management techniques are new processes that are in development at airports, or have the potential to be developed in the future and will be discussed briefly below.

3.3.1. Outbound Holding

The idea of outbound holding is to "swap" an aircraft that is waiting for departure from the gate with an inbound aircraft that would like to use that same gate. It is a new process that Schiphol is currently exploring, but that has not been tested extensively before. The reason for the introduction of outbound holding is to try to prevent the double holding delay, as discussed in [subsection 2.2.4](#). The technique is therefore interesting to explore further this study that focuses specifically at apron operations. The procedure currently in development at Amsterdam Airport Schiphol was discussed with Schiphol experts and is the following:

1. Gate planning notices double holding, i.e. outbound aircraft waiting for departure from gate and inbound aircraft waiting for the same gate.
2. Gate planning checks the following minimum conditions:
 - Did the inbound flight arrive much earlier than its scheduled arrival time?
 - Does the outbound flight need de-icing?
 - Are there low visibility conditions at the airport?

If any of these conditions apply, outbound holding cannot be arranged according to the existing protocols.

3. Check if the target start-up time (TSAT) of the outbound flight is 15 minutes after the last known target off-block time (TOBT). If this is the case, go to step 4, otherwise outbound holding cannot be arranged according to existing protocols.
4. Gate planning makes an estimation of the potential maximum remote holding time of the outbound aircraft. Based on subjective judgement, it is decided whether this value is within acceptable limits. If this is the case, the decision to arrange outbound holding is made.
5. Together with the ground handler, it is decided whether towing the outbound aircraft to the remote holding position is possible. There are two options from this point:
 - **Towing is possible:** This is the preferable option. In this case the outbound aircraft is towed to the remote holding position. As discussed before, towing traffic is under the control of Apron Control, which means close collaboration between Apron Control and ATC is required.
 - **Towing not possible:** In this case, the aircraft will start its engines after push-back and taxi to the remote holding position on its own power. This is controlled by ATC like regular taxiing traffic.
6. Once the gate is vacated by the outbound aircraft, ATC guides the inbound aircraft to its gate.

Locations that can be used for outbound holding are the positions on the P-apron, positions on the R-apron and P6 and P7. Restrictions and locations of these positions were previously discussed in [subsection 2.2.4](#). In theory, applying outbound holding is beneficial for the overall operation, as the total waiting time is reduced. An especially interesting case is when TaxiBot is used for towing the outbound aircraft to the remote holding position. In this case, the movement of the outbound aircraft and the waiting time at the remote holding positions will result in less fuel burn than the situation in which the inbound aircraft or the outbound aircraft has its engines on.

However, the concept has not been tested before and the system wide implications are therefore still unknown both with and without TaxiBots. One of the aims of this thesis is to contribute to this lack of knowledge.

3.3.2. Other strategies

If a concept of TaxiBot is implemented at Schiphol, the taxi operation changes significantly. An autonomous version of the TaxiBot changes the operation even more. This might also require and offer the opportunity for the development of new techniques in order to manage the traffic over the airport. Although an ongoing study [90] focuses on the location of decoupling points and the traffic flow around these points, studies that focus on the need for and the development of new traffic management techniques around the apron and the gates have not been performed before. This is another area where this research will add to the body of knowledge.

3.4. Concluding Remarks

From the developments discussed in this chapter, it can be concluded that the conventional operation could change significantly due to the introduction of new concepts. The A-SMGCS system is already being implemented at European airports and is an state-of-the-art example of a shift from centralized control to more distributed control. Although forming a first architecture for this, limited research has been performed that explores the techniques required to successfully implement the routing and control service. Engine-off taxiing techniques discussed in this chapter are further away from being implemented on full-scale in the operation. From the alternative taxiing methods, TaxiBot, an external towing vehicle, was chosen to be considered

in this research. Reasons that contribute to this are the fact that it is currently the only certified concept on the market and that Amsterdam Airport Schiphol is already experimenting with this concept. Although conceptual tests have been performed and first experiments have been conducted, these concepts have not been considered on a full-scale level extensively. Previous studies from an industry perspective have only focused on implementation of TaxiBot in the current system without adaptations in the control methods. A single exploratory academic study provided first insights in the concept of autonomous TaxiBots based on distributed coordination and planning, but excluded the aprons and gates from the model. An important part of the operation that includes apron operations and holding was therefore ignored.

Having introduced the developments in airport ground surface movement operations, the next section will discuss how these operations and new concepts of operations can be modelled.

4

Modelling Ground Surface Operations

The previous chapters have introduced the current ground operations at Amsterdam Airport Schiphol and the developments that are taking place in this field. Where the topics discussed up to this point were mainly focused on the design of the actual ground surface operation, this chapter will focus on the research that has been done in the area of modelling this operation and conclude with a research gap.

As discussed in the previous sections, the conventional airport grounds surface operation is set up with many actors and interactions between them. The control over the conventional ground surface operation is organized in a centralized way, where control and coordination of the traffic at the airport is the responsibility of Air Traffic Control (ATC). Such a centralized network is visualized in [Figure 4.1 \(a\)](#). The difficulty and workload resulting from this set up is demanding for Air Traffic Control Officers (ATCO's), especially on busy airports with complex structure. Another network structure is the decentralized network, visualized in [Figure 4.1 \(b\)](#). The set up of the ATC system as a decentralized system is beneficial as processes can be broken down in smaller and less complex problems. Finally, the distributed network, visualized in [Figure 4.1 \(c\)](#) is another alternative, where the control responsibility is spread out over the entire network.

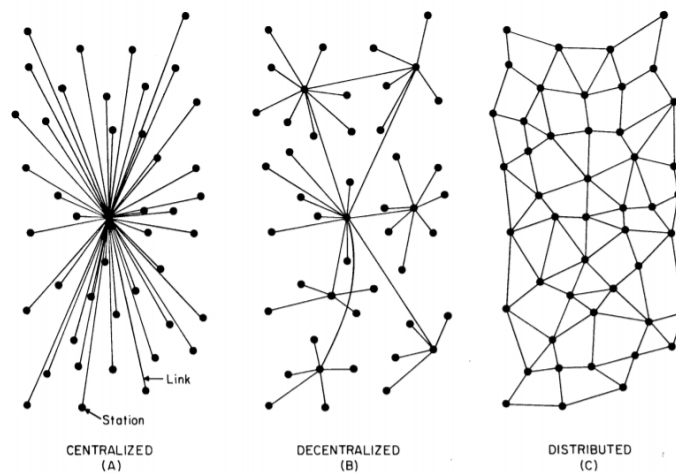


Figure 4.1: Centralized, Decentralized and Distributed Network [7].

The interactions in the airport ground surface operation are of a more distributed nature. As shown in the socio-technical system in [chapter 2](#), there are many stakeholders that communicate and interact with one another. Different modelling techniques to model such a complex system will be discussed briefly in the next section, after which the agent-based modelling methodology will be described in more detail. Then, agent-based models that have been developed in the Air Transport and Operations group at Delft University

of Technology will be discussed with their main conclusions and shortcomings. Finally, a research gap will be identified based on the conclusions from this section and the previous subsections.

4.1. Complex Systems Modelling

In this section, several modelling techniques will be evaluated on their suitability to model the airport ground surface movement operation, or in general, a complex system. A choice for the most suitable modelling technique for this research will be made based on the advantages and disadvantages of each technique. First, linear programming and search algorithms will be discussed. Then analysis of the operation based on historical data will be discussed and the modelling the operation with a Markov Chain will be considered. Finally, Agent-Based Modelling (ABM) will be discussed.

4.1.1. Linear Programming and Search Algorithms

Linear Programming (LP) is a technique in which a certain objective function is usually maximized or minimized by determining optimal values for the decision variables. Constraints are specified beforehand that have to be respected in the final solution. The solution to a LP problem is a set of values for the decision variables that result in an optimal solution when filled in into the objective function. Although the fact that the solution to a LP problem is optimal is an advantage of LP, the method also has several disadvantages. The first is that it solves the problem in a centralized way, which means that interactions between agents and between agents and their surroundings cannot be accounted for. In the context of airport ground surface movement operations, this would for example automatically introduce the assumption that aircraft follow their planned routes perfectly and that collisions between aircraft do not occur. A second disadvantage of LP was noted by Benda [9] who refers to Atkin et. al [5] stating that LP solvers might be impractical for large problems due to their high computational time needed to find a solution.

Search Algorithms are less prone to long running times, as they look for solutions that are close to optimal instead of optimal solutions. Search Algorithms were considered by Benda [9] and Noortman [55] for application in airport ground surface movement operations, who found that search algorithms have similar disadvantages as LP methods in the sense that they optimize from a central point of view. Thereby assume that plans are executed perfectly.

4.1.2. Historical Data

Another method to model airport ground surface movement operations is by analyzing historical data. This data is then used to make predictions about the future. Benda [9] evaluated methods that use historical data for modelling airport taxiing operations. He found that, although these methods can provide accurate predictions of for example taxi times when variations in the data are small, they are more challenging to use when variations in the data are larger. Besides that, Benda notes that interactions and disruptions are usually not accounted for in these techniques. Another disadvantage of these techniques is that it is relatively difficult to test new scenarios for which limited data is available. The effects of having more traffic at the airport or the effects of the introduction of electric taxi solutions at the airport are therefore more challenging to analyze with such a model. Also the evaluation of effects of new techniques such as outbound holding are difficult to analyze with this method.

4.1.3. Markov Chains

A Markov Chain is a stochastic model in which the next state is independent from previous states. This is also referred to as the memoryless property. Benda [9] evaluated approaches that use Markov Chains for modelling airport ground surface operations and found that the combination of Markov Chains and Reinforcement Learning is commonly used to estimate taxi times. From his evaluation, Benda concludes that this combination of Markov Chains and Reinforcement Learning is less suitable when there is a high variation in taxi times. In the case of Schiphol, with one runway (the Polderbaan) located far from the other runways, variations in taxi times can be large and this therefore makes this method less suitable, which is in line with the conclusion by Benda. Besides that, the memoryless property suggests that past events cannot be accounted for when using Markov Chains. In the actual airport ground surface operation it might however be desirable to take the past into account, for example, an aircraft might be given priority when it has already been waiting for a long time.

4.1.4. Agent-Based Modelling

A method that is commonly used to model systems with many interactions is agent based modelling (ABM). It is considered as a method that is especially suitable for modelling complex systems [85]. The main features of agent based modelling were discussed by Sharpanskykh [85], which will be used as a baseline to describe the technique. A characteristic of agent-based modelling is that it takes a bottom up approach to model a system rather than a top down approach. In this way, a system can be defined by looking from the perspective of heterogeneous intelligent individual components and the interactions and dependencies between them. This makes the technique especially suitable to specify complex systems in a natural way. Also uncertainty and system dynamics can be captured well in an agent based model, which was not the case in many of the models discussed previously in this section. Examples of problems that have been modelled with help of an agent-based model are natural phenomena such as bird flocking [61] or human systems [12]. Besides that, transport problems such as order picking in warehouses have been modelled with an agent-based model [50] and many applications of agent-based modelling can be found in the aerospace context. Examples are modelling (non-)autonomous taxi operations [9] or autonomous baggage handling systems [18]). After developing an agent-based model, it can be used effectively to analyze emergent behavior in a system. Although this means that the technique offers many benefits over other modelling techniques, a downside is that the development of the model can be time consuming. For this study, the benefits of the technique outweigh this disadvantage, as the interactions between all actors are of major importance. Agent-based modelling is therefore considered to be a suitable and effective modelling technique to model the airport ground surface operation with its many actors.

Elements of an ABM

The main elements that specify a multi-agent system are the environment, the agents and the interactions between agents and between agents and the environment. Each of these elements will be discussed below.

1. Environment: The environment of a multi-agent system consists of all the elements in a system that are not an agent. Agents can observe or act upon the elements in the environment. Environments can be characterized as ([87], [64]):

- **Deterministic vs. Non-Deterministic:** In a deterministic environment, the result of any action has a guaranteed effect. In a non-deterministic environment, there is no certainty about the result of an action.
- **Static vs. Dynamic:** Although a static environment does not change by itself, agents can change a static environment. A dynamic environment cannot be considered to remain unchanged over time.
- **Accessible vs. Inaccessible:** An agent in an accessible environment is able to obtain complete, accurate and up-to-date information about the state of the environment.

2. Agents: An agent is an autonomous actor with two characteristics. It is able to perceive its environment through its sensors and that acts upon its environment with its actuators [64]. For each agent in an agent-based models, a set of rules can be defined describing the behavior of the agent. This can be based on both the agent's perception of the environment or on the internal states of the agent [87]. A system with multiple agents is referred to as a multi-agent system.

3. Interactions: In a multi-agent system, agents can interact with each other and with the environment. This enables agents to work together (cooperate) or to compete. Cooperation and competition are both examples of coordination in a multi-agent system, which will be described in the next section.

4.2. Coordination in Multi-Agent Systems

This section will briefly discuss why coordination is required for airport ground surface movement operations. Two ways of coordination will be described. These are cooperation and competition, as shown in Figure 4.2.

A non-coordinated system with agents moving over the airport could lead to collisions and inefficient use of the airport ground surface. In the current system, Air Traffic Control Officers (ATCO's) therefore coordinate the movements of individual agents over the airport. This can be done in both a cooperative and a competitive setting. In a cooperative system, agents work together so that they can achieve something which they cannot achieve on their own. An example is that agents share information about their intentions accurately

and timely, so that a planning can be made in which all agents reach their goals safely and efficiently. In a competitive system on the other hand, agents try to achieve something that can only be achieved by a part of the group of agents, thereby maximizing their own benefit or the benefit of a few [87]. An example would be that aircraft carriers bid to get a departure slot that is only available for one of them.

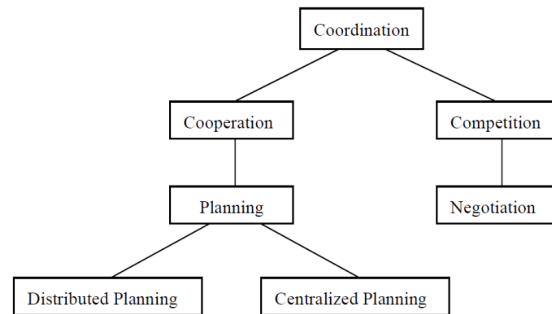


Figure 4.2: Different forms of coordination in multi-agent systems. Obtained from [87].

For this research, multi-agent planning is considered. The aim is to benefit the efficiency of the entire airport system and not benefit a few vehicles moving over the airport. Six different phases in multi-agent planning are described by Weerd et al. [15] based on Durfee [19] and are shown below:

1. **Global goal and task refinement:** The goal or task are split into smaller sub tasks that can be assigned to agents.
2. **Task allocation:** The sub tasks resulting from step 1 are assigned to agents.
3. **Coordination before planning:** Conflicting plans can be avoided by the definition of rules and constraints before individual plans are made.
4. **Individual planning:** Individual agents plan so that they can reach their goals. Coordination during planning can be part of individual planning.
5. **Coordination after planning:** Given the plans of individual agents plans are coordinated.
6. **Plan execution:** Plans are executed and results of sub tasks are synthesized. Plans can be coordinated during the plan execution phase.

Weerd et al. [15] describe that not necessarily all of these phases are part of the planning. Task allocation is for example not necessary if two agents both have to be routed over the airport. Sharpanskykh [86] adds a seventh step of "replanning" to the phases of planning. This refers to the repetition of step 4,5 and 6 in case disruptions in the system require the planning to be updated.

Although Air Traffic Control Officers (ATCO's) and the predefined standard taxi directions coordinate the movements of individual agents over the airport in the conventional operation, multi-agent planning and scheduling algorithms can potentially be used to do this autonomously. Previous research that considered the application of such techniques will be discussed in the next section. Multi-agent path planning techniques will then be discussed in more detail in [chapter 5](#).

4.3. Agent Based Modelling of Airport Ground Surface Operations at ATO

Now that the agent-based modelling method has been introduced, previous research that applies agent-based models in the context of airport ground surface operations will be discussed. An active research line at the ATO department of the TU Delft is dedicated to the agent-based modelling of ground surface operations in which multiple PhD students and MSc students have developed parts of a simulator of Amsterdam Airport Schiphol ground operations. An initial version of an Agent Based Modelling simulator for airport ground surface operations was developed by Heiko Udluft during his PhD thesis [97]. This simulator was later used and adapted by MSc students T. Noortman [55], K. Fines [29], B. Benda [9] and S. Polydorou [59] during their thesis work with the application on Amsterdam Airport Schiphol. This work was all produced under supervision of dr. O.A. Sharpanskykh as part of one of his research lines: Multi agent, distributed control of air transport operations. The main characteristics of the models used by these students are summarized in [Figure 4.3](#) and their main findings will be discussed in more detail in this section.

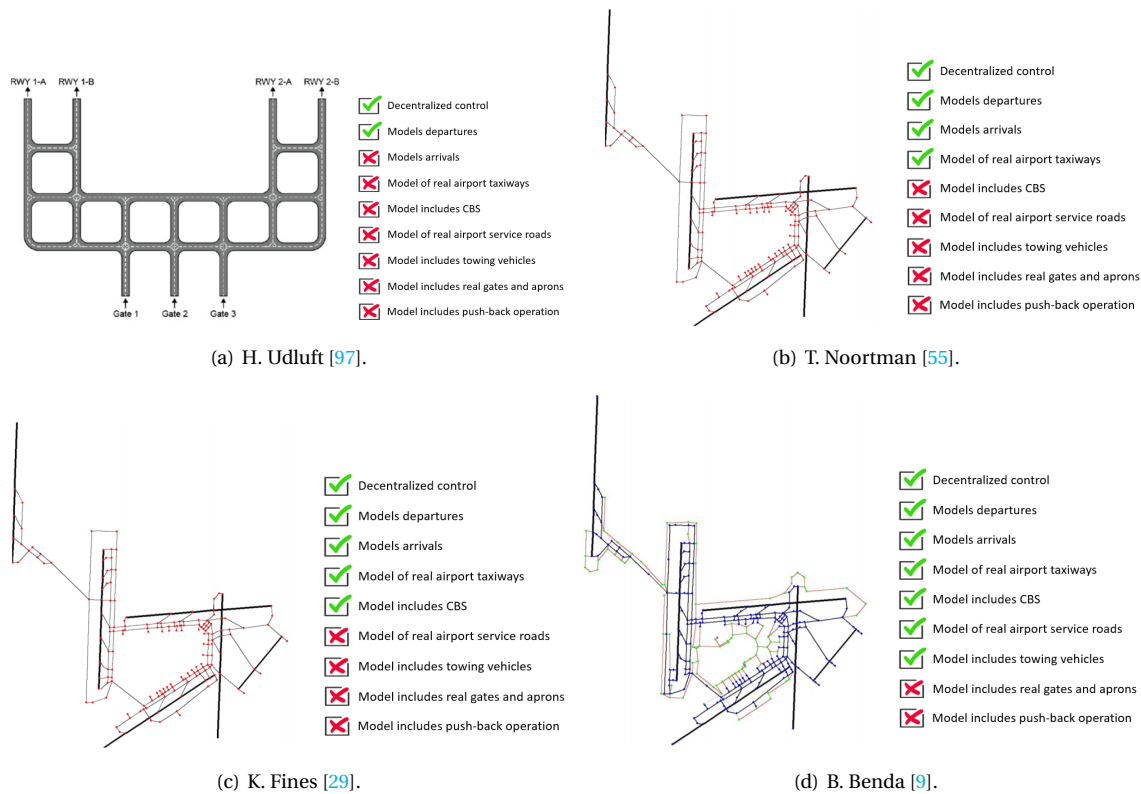


Figure 4.3: Comparison of previous studies at ATO. Adapted from Soomers [90].

4.3.1. Heiko Udluft: Decentralization in Air Transportation

During his PhD work at the TU Delft, Udluft focused on the decentralization in air transportation. It was during this study that an initial version of the agent based modelling simulator was developed, as described in Udluft's PhD thesis [97]. Simulations with the agent based model were performed using an airport layout shown in 4.3(a) with 3 gates, 2 runways and 2 runway entry/exit points per runway. At each intersection of two taxiways, an intersection agent was placed which gives instructions to aircraft approaching this intersection agent in the taxiway network. The intersection agents are responsible for the planning and coordination of the aircraft routes in the system. Intersection agents can then locally instruct aircraft to follow a specific heading or to decelerate or stop in case another aircraft agent is already occupying the next edge the aircraft has to pass. Aircraft agents are supposed to follow up the instructions given by intersection agents and at the same time, they have to maintain a safe separation distance between them and other aircraft in the network.

Coordination between intersection agents by means of an auction mechanism was investigated. The intersection agents make bids on specific segments of the taxiway network based on the interests of the aircraft under their command. The highest bidder wins and is then assigned the right to taxi over the taxiway segment. One of the conclusions of the study was that coordination improves the performance of decentralized control for taxiing aircraft. Recommendations from the study included the advice to test the concept of decentralized control with real airport layouts. This was started by T. Noortman in his MSc thesis.

4.3.2. T. Noortman: Agent-Based Modelling of an Airport's Ground Surface Movement Operation

Noortman [55] conducted a case study at Amsterdam Airport Schiphol. A layout of the airport was constructed using the data of the location of taxiway segments. This layout is shown in 4.4(a). In order to make the layout suitable for use in the simulation model, Noortman decided to simplify the layout to a layout shown in 4.4(b). A trade-off was made between the level of detail and the computational time which is why several segments in the model were merged. Examples are corners consisting of multiple segments, straight roads consisting of multiple segments or segments located very close to each other.

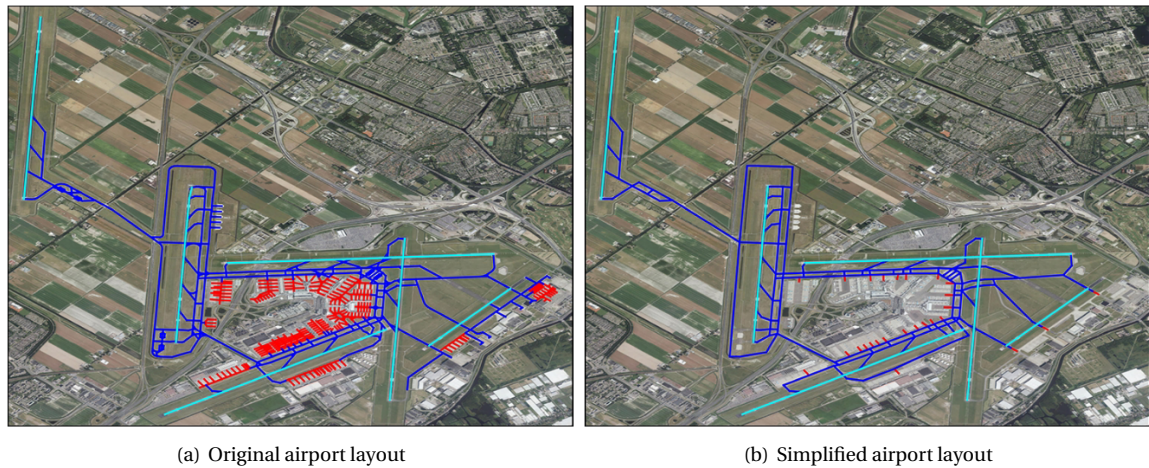


Figure 4.4: Effect of simplifying the airport layout [55].

The focus of the study by Noortman was on the taxiway network, which is why the apron operations were excluded from the main study. One of the assumptions of his study was therefore that gates and aprons could be merged and represented as one single meta-gate agent in the model, as shown in Figure 4.5. Towing operations were excluded from the model and it was assumed that aircraft spawn at the merged gates, the meta-gate, following a predefined schedule.

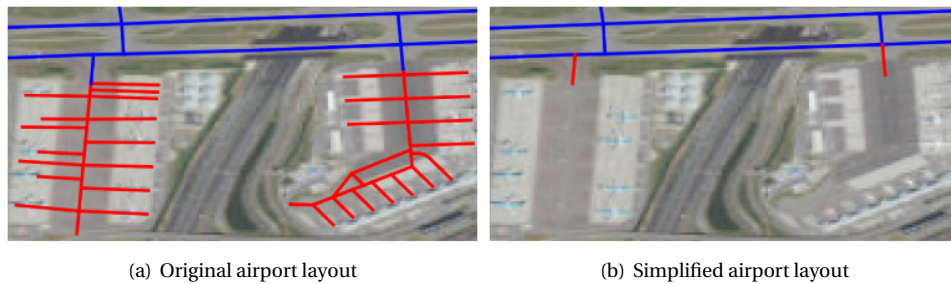


Figure 4.5: Effect of replacing an apron by a single entry/exit node [55].

An example of merging a cluster of nodes located close to each other is shown in 4.6(a). As can be seen in the image, the simplification limits the entry/exit possibilities to and from the apron.

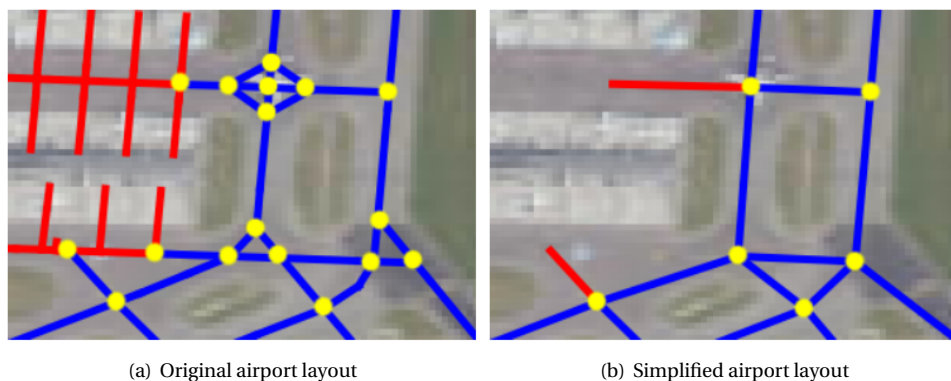


Figure 4.6: Effect merging a cluster of nodes to one node [55].

One of the main recommendations from Noortman's study was to include the apron and gate operations in future research. The main reason for this was his comparison of the simulated operation with the actual

operation. In reality, some arriving aircraft were sent to a remote holding position if their gate was not available, which could not be included in the model without the gate and apron operations included. Besides that, Noortman noticed that, in reality, push-back from some gates will block parts of the main taxiways temporarily which might have an effect on the traffic density on the airport. This second point was not simulated in his model either as push-back operations were not included.

4.3.3. K. Fines: Agent-Based Distributed Planning and Coordination for Resilient Airport Surface Movement Operations

Fines [29] continued the research in the area of distributed airport ground surface operations. Fines investigated the potential for decentralized control at airports under adverse conditions. His research especially focused on the disturbances caused by runway configurations, occurring multiple times a day due to changing weather conditions and the resilience of the airport systems that coordinate the ground surface operations movements.

A case study was conducted based on the decentralized control system called Follow the Greens, a project that is being tested and further developed in the Single European Sky ATM Research (SESAR) program and previously discussed in this literature study. The previous work done by Udluft and Noortman was further developed to be used in a combined case study of the Follow the Greens concept and decentralized control at Schiphol airport.

Fines was the first student at ATO to implement a version of the conflict based search (CBS) algorithm in the simulator as a method for conflict avoidance and path planning. This algorithm will be discussed in [subsection 5.2.2](#) in detail. The results from Fines' experiments show that the distributed planning and coordination mechanisms contributed to the resilience at the airport [30]. Taxi times after a changed runway configuration in the simulated version were compared to taxi times in the real life situation and were found to be equal or better [30].

Apron and gate operations as well as towing and push-back operations were still excluded from the model used by Fines. This means that the shortcomings that were identified by Noortman, such as the lack of simulating remote holding and push-back operations are still part of the model. Main recommendations from Fines' study include the advice to improve the predictive and anticipatory elements in the model. He observed that the actors in the model react to runway re-configurations only at the moment they occur, while in reality this change is already known some time in advance. Besides that, Fines' noticed that the prediction of taxi times, used in the forward simulation by CBS, can be improved to be more accurate.

4.3.4. B. Benda - Agent-Based Modelling and Analysis of Non-Autonomous Airport Ground Surface Operations

Benda [9] continued on the work of Fines. He did an exploratory study on the effects of the implementation of TaxiBots at the current ground operations at AAS. Service roads were included in the layout of Amsterdam Airport Schiphol to route these additional vehicles when they were not attached to an aircraft. This increased the number of vehicles in the simulator significantly, as next to aircraft agents, TaxiBots now also had to be routed, sometimes crossing or driving on taxiways. Similar to Fines, Benda used a version of CBS for path planning and conflict resolution.

Although TaxiBots were included in this research, the assumption to exclude push-back and apron and gate operations was also made. Benda also recommended to include apron operations in future work so that the realism of the model can be improved and so that conventional towing and taxi operations can be compared with the novel concept of TaxiBot towing more accurately. Benda also identified the potential for the inclusion of runway scheduling by including gate operations. This would enable the possibility of releasing aircraft from the gate at a specific time, possibly by including gate holding, rather than releasing them based on a fixed schedule. The potential for improving taxi time predictions as part of the forward simulation used by CBS was shared by Benda following the recommendation by Fines.

4.3.5. Other ongoing studies at ATO

At the time of writing this literature study, two partial results from other MSc Thesis students were available in the form of a literature study and a research plan. S. Polydrou [59] focuses on improving the estimation of taxi times in the model by means of supervised learning. J. Soomers [90] focuses on the locations of the uncoupling of TaxiBots in an autonomously coordinated multi-agent system in order to study the ground delays.

4.4. Concluding Remarks and Research Gap

After an initial study considering the potential for decentralized control in air transport operations [97], several studies have considered the potential for distributed and decentralized airport ground surface operations using a real airport layout of Amsterdam Airport Schiphol as a case study ([29], [30], [55]). Also the inclusion of TaxiBot operations has been studied on a similar airport layout as used in previous studies, extending it with service roads [9]. Although these studies have all contributed to an improved insight in how distributed planning and coordination mechanisms can be used to control traffic, they have all used a simplified layout of the airport excluding the apron and gate areas. This simplification raises the restriction that push-back operations, runway sequencing and remote holding cannot be evaluated in these models. These operations do however have a large effect on the performance of the operation. Besides that, the introduction of TaxiBot operations is claimed to increase efficiency at the platforms as engines are not started at this point anymore. This is therefore one of the locations where the novel concept of TaxiBot towing will be considerably different from the conventional operations. However, it has never been studied in detail in the current simulator implementation.

Case studies with a single TaxiBot were conducted at Schiphol in 2020. Additionally, TO70 performed a simulation study for Schiphol evaluating the implementation of TaxiBots in the current system. The feasibility of full Taxibot implementation for towing of outbound flights was found to be infeasible in its current form but feasible under the condition that solutions would be found for main restrictions identified in this study. From an ATC viewpoint, capacity, on time performance and workload would be negatively influenced in the scenario that was studied. Main points that need improvement are related to the control of TaxiBots and managing delays. Although these industry studies have provided valuable insights, both make important assumptions. The case study was only performed with a single TaxiBot which means the effects of TaxiBot operations on a global level could not be evaluated properly. The simulation study that was performed does provide valuable insights on a global level but assumes implementation in the current system thereby disregarding potential future developments in the distributed or decentralized control in the airport ground surface operation.

An area that has not been studied thoroughly is the implementation of a novel taxi concept of TaxiBot towing in a system using distributed control mechanisms, while also including the apron and gates in the airport layout. Inclusion of apron and gates enables the analysis of important processes such as gate holding and remote holding, push-back and runway sequencing. Detailed descriptions of the apron and gate operations are available, including timing data of aircraft arriving and leaving the gate. This makes the development of a model including this additional area of the airport both feasible and valuable in the research field of airport ground surface movement operations. This research therefore aims to analyze a novel taxi concept of electric taxiing operations using a distributed multi-agent approach with a focus on apron operations.

5

Multi-Agent Planning & Scheduling

In [chapter 2](#) the conventional ground surface operation was discussed and in [chapter 3](#) the developments in this field were outlined. It was shown that these developments, together with the increased traffic at the airport form a new concept of operations. Previous research at ATO, discussed in [chapter 4](#), showed that a concept of distributed multi-agent ground surface operations was effective in controlling traffic over the airport's ground surface. These studies did however exclude gates and aprons from the model which made it impossible to study the effects of apron and gate related processes on the operation. The evaluation of traffic management techniques such as gate holding, remote holding or push-back rate control were therefore not studied before in an distributed controlled system. The concept of TaxiBot, as discussed in [chapter 3](#), will change the operations at the airport significantly as well. A main difference with the conventional operation is the engine-start that does not take place at the apron anymore but close to the runway. Although several studies have been done or are currently going on into the concept of autonomous TaxiBot operations, complete system-wide evaluations that include the processes at the apron were not available. The combination of distributed multi-agent airport ground surface operations on an extended airport layout with aprons and gates and the concept of autonomous TaxiBots is therefore interesting to study further.

As discussed before in this literature study, actors at the airport share a layout with limited space. There is for example only a limited capacity at gates and take-offs and landings are sequenced by ATC to satisfy certain constraints. In [chapter 4](#), different forms of coordination in multi-agent systems were discussed with a distinction between cooperative coordination and competitive coordination. For this research, the airport operations are considered as a system with cooperative actors that, together, try to achieve what they cannot achieve on their own. This extends the problem of planning for one single vehicle movement to the the problem of planning for multiple vehicle movements. The combination of this classical single agent planning for multiple robots and coordination is referred to as multi-agent planning.

The first part of this chapter will discuss single agent path planning and basic techniques to solve this problem. The extended problem of multi-agent path planning, where paths of multiple agents have to be planned and coordinated will be discussed in the remainder of the chapter. The multi-agent path finding (MAPF) problem and the generalized multi-agent motion planning (MAMP) problem will be discussed including the available techniques to solve these problems. The most suitable technique to be applied in this research will finally be chosen in a trade-off.

5.1. Background on Path Finding

This section will give an introduction to the shortest path finding problem and then generalize this problem to the multi-agent path finding problem (MAPF).

5.1.1. Shortest path finding with A*

Before considering the problem of multiple moving agents over the airport, the simpler problem of one single moving agent moving over the airport will be discussed. An airport layout can be converted to a graph layout with edges representing taxiways, aprons and runways and nodes representing gates, intersections and run-

way entries and exits. This problem can then be solved similarly to a shortest path problem, on a graph with edges and nodes. Well-known approaches to solving this problem are the Dijkstra's algorithm [17] or the A* algorithm [33]. The A* algorithm will be introduced in the beginning of this section as it forms the basis of many multi-agent path planning techniques that will be discussed later in this chapter.

A* is a search algorithm that finds a shortest path from a start node to a goal node. It uses a graph layout represented as $G(V, E)$ with nodes (V) and edges (E) between these nodes that have an associated weight. The algorithm then uses three different parameters. It considers the node with the lowest f -score from the open set and saves this as the current node. Then, if this node is not the goal node, the algorithm explores the neighbors of the current node. $g(n)$ is the cost of moving from the start node to a neighbor node n and $h(n)$ is the heuristic value. This heuristic value is an estimation of the shortest path from the neighbor n to the goal node. The heuristic should be admissible, which means that the estimation is never an overestimate of the actual cost to reach the goal node. Finally, $f(n) = g(n) + h(n)$ is the evaluation function or f -score, which is updated for neighbor node n . In the next step, the algorithm then considers the new node with the lowest f -score as the current node and repeats the above steps. A* is proven to be optimal and complete, meaning that is guaranteed to find a shortest path from the start node to the goal node [33].

The heuristic value h used in the A* algorithm makes it more efficient than the Dijkstra algorithm that does not use an heuristic. A commonly used value is the Euclidean distance, which is the straight-line distance between two points and therefore never overestimates the cost. The most accurate heuristic is the actual shortest path to the goal node, but when searching from the start node, this actual shortest path cannot be known (note that this heuristic can be computed before running the algorithm by doing a search from all goal nodes to the start nodes).

5.1.2. Multi-agent path finding (MAPF)

In the multi-agent path finding (MAPF) problem, a path has to be planned for multiple agents from their start positions to their goal positions. Collisions between agents cannot occur in a solution to the problem. Agents can either wait or make a movement in each timestep in the classical problem, which means that a solution consists of collision free paths if two agents are not at the same node at the same time. Although different optimization objectives can be used, a commonly used objective is to minimize the makespan. An optimal makespan solution is one in which the total number of steps necessary for all agents to move from their start positions to their goal positions is minimized [50]. Applications of the multi-agent path finding problem and terminology will be discussed below, after which an example MAPF problem will be discussed.

The MAPF problem has many real life applications. There are examples of warehouses where paths for multiple robots are constantly planned so that they can pick up and deliver packages without colliding with other robots [36]. Besides that, the MAPF problem is relevant for the routing of multiple autonomous road vehicles which move from location to location. Applications within the aerospace industry are already found in the use of multiple intelligent robots for baggage handling [99], where robots are routed automatically while they move bags or in the application of routing novel autonomous towing vehicles over the airport's ground surface without collisions ([9], [54]).

An unified terminology for MAPF was proposed by Stern et al. [94] that will be used as the baseline for this subsection. As discussed before, collisions between agents cannot occur in the solution to a MAPF problem. In a more general form, collisions can be considered to be a type of conflict. Five general types of conflicts can be described, as shown in Figure 5.1. An "edge conflict" occurs when two agents are planned to traverse the same edge in the same direction at the same time step. When two agents are not planned to traverse the same edge, but when they are planned to be at the same vertex at the same time, a "vertex conflict" occurs. Agents that are following each other are in a "following conflict" when an agent is planned to be at the vertex at which another agent, travelling in the same direction, was in the previous time step. A "swapping conflict" occurs when two agents are planned to traverse the same edge at the same time in the opposite direction. The last conflict described in [94] is a "cycle conflict", which occurs when multiple agents move in a cycle and for every time step, the previous position of an agent is occupied by another agent. As will be discussed later in this section, three types of conflicts are added to this set when agents can occupy more than one edge or vertex at each point in time.

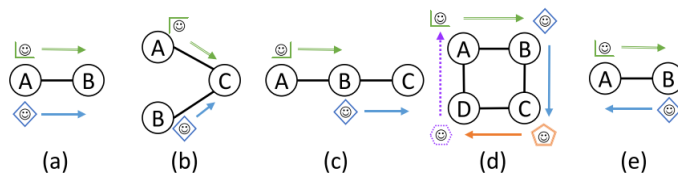


Figure 5.1: Types of conflicts. (a) edge conflict, (b) vertex conflict, (c) following conflict, (d) cycle conflict and (e) swapping conflict [94].

Example MAPF problem

An example of the MAPF problem, as discussed by Sharon et al. [84], is shown in Figure 5.2. In this example, two mice (agents in this example) start at positions S_1 and S_2 . Their goal is to reach the cheese at their goal locations G_1 and G_2 respectively with the optimization objective to minimize the makespan. If the mice would be in the network alone, they would be able to reach their goal in three time steps (Agent 1: $[S_1 \rightarrow A_1 \rightarrow D \rightarrow G_1]$ and Agent 2: $[S_2 \rightarrow B_1 \rightarrow D \rightarrow G_2]$). When they move at the same time however, these agents will collide at D when the agents will be in a vertex conflict. One of the agents therefore has to wait for one time step in order to avoid the collision. An example of an optimal solution is the combination where agent 1 follows the path $[S_1 \rightarrow A_1 \rightarrow D \rightarrow G_1]$ and agent 2 waits for one time step at node B_1 to avoid collision at node D and thus follows the path $[S_2 \rightarrow B_1 \rightarrow B_1 \rightarrow D \rightarrow G_2]$.

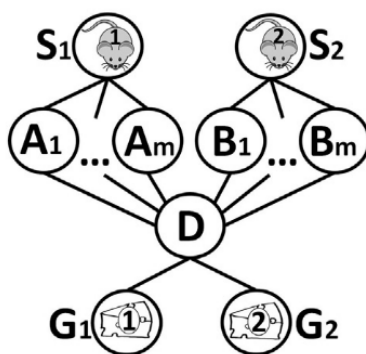


Figure 5.2: Example MAPF problem where two mice move from their start positions S_1 and S_2 to their goal positions G_1 and G_2 [84].

Having introduced the multi-agent path finding problem, the next section will first discuss existing approaches to solve this problem. First, approaches that assume discretized time, perfect plan execution and synchronized movements of agents will be discussed. However, these assumptions do not always hold for real world robots as discussed in section 5.3, which is why section 5.4 will cover mechanisms that take into account more realistic robots in or after planning.

5.2. Classical approaches for MAPF

This section will discuss existing, more theoretical approaches for solving the multi-agent path finding problem. These algorithms assume, in their original descriptions, that agents can perform a wait action or move action in each time step with a duration of exactly one time step. In addition, these approaches assume that the agents do not have any geometrical shape and therefore do not block any other part of the graph except the node they are on, or the edge they use to move. Those approaches are referred to as "classical MAPF" approaches for the remainder of this section. Modifications to these algorithms can often be made to relax some of these assumptions and will therefore be discussed after introducing the original variants.

Sharon et al. [84] and Stern [93] provided an overview of available multi-agent path finding techniques in their work. They compared different algorithms on the theoretical side, reviewing completeness, optimality and efficiency (time needed to find a solution). Recent MSc students from Delft University of Technology reviewed MAPF approaches, including those reviewed by Sharon et al. and Stern, for their suitability to be

applied in the context of airport ground surface operations ([29], [59] [90]). The results from these reviews are considered as valuable for this research, as the context of application is very similar. A general overview of the work of these MSc students and the reviews by Sharon et al. and Stern will therefore be given in this first section and will be used to narrow down the available MAPF algorithms to the most suitable ones for application in this research. Besides that, algorithms that were not considered in previous literature studies will be evaluated.

Several groups of algorithms are considered as unsuitable or inferior to other approaches for the application in airport ground surface operations. One of them is the family of rule-based approaches, that work based on a set of pre-defined movement rules. Although these approaches do usually find a solution relatively fast due to the fact that they do not perform a search, their solution quality is relatively low ([59], [84]). Polydrou [59] refers to the Push and Swap [47], Push and Rotate [16] and Tree-Based Agent Swapping Strategy [43] as rule-based approaches that include 'swap-operations'. For this swap operation, the algorithms assume that agents are capable of moving in all directions. Polydrou noted that this is challenging to implement at many airports, as the places where aircraft can turn 180 degrees are limited and thus swapping movements are often impossible. This set of rule-based approaches are therefore considered as inferior to other approaches that will be discussed in this section and will not be considered further for application in this study.

Another group of approaches for MAPF is the group of reduction based approaches in which the MAPF problem is translated into a set of simpler problems that are then solved. Boolean satisfiability (SAT) [95], Answer Set Programming (ASP) [21], Integer Linear Programming (ILP) [102] are examples of such approaches [84]. Sharon et al. [84] point out that these reduction based approaches are inefficient for solving large MAPF problems as they take a lot of time to find a solution. Polydrou [59] also questions the suitability of reduction based approaches in the context of airport ground surface operations due to their high running times and because approaches have not been compared to other state-of-the-art MAPF solvers. For these reasons, it is decided that these approaches will not be considered any further for this study.

Approaches that will be reviewed in more detail in this first section are divided into three categories in this literature study. Extensions on A* will be discussed in [subsection 5.2.1](#), then CBS-based approaches in [subsection 5.2.2](#) and finally priority based approaches in [subsection 5.2.3](#). Note that additional approaches, focused on real-world robot path planning will be discussed in the next section. These latter approaches are often based on existing traditional approaches discussed in this section.

5.2.1. Extensions of A*

The standard version of A* as described previously in this chapter is very inefficient in finding a solution for the MAPF problem. The main reason for this is that size of the state space grows exponentially with the number of agents and the branching factor of the states may also grow exponential with the number of agents [84], [93]. Sharon et. al [84] illustrates this by giving the example of 20 agents that have the option to move up, down, right or left or wait which would already lead to 5^{20} neighbor states. Several ways have been proposed to make A* suitable for solving the MAPF problem. In this section, variants of the A* algorithm will be discussed that extend the A* algorithm to make it suitable for MAPF. Other approaches of adapting A* to make it suitable for MAPF introduce priorities and split the problem into single agent path finding problems. These will be excluded from this first section and will be discussed later in [subsection 5.2.3](#).

Sharon et al. [84], Stern [93] and Polydrou [59] have reviewed the extensions of A* in their work and highlighted several approaches to improving A* for MAPF. First they refer to two improvements by Standley [92] namely Operator Decomposition (OD) and Independence Detection (ID), which will be discussed briefly below.

The aim of the Operator Decomposition (OD) framework is to reduce the branching factors of the states in A*. Standley notes that the framework can be used in other search algorithms as well for the same purpose. OD subdivides each time step so that it only considers one agent at a time, following a random and fixed ordering. When it considers one agent, OD expands an intermediate node that is then used to expand new intermediate nodes (by placing it in the OPEN list). Only when considering the last agent, a regular node as in A* is introduced. The result of this approach is that the branching factor is significantly reduced to the branching factor of one single agent instead of the exponential branching factor in A*. Besides that, the number of regular nodes generated without being further expanded (because the f -score is larger than the optimal solution) is

lower than it would be in A^* . Based on the fact that A^* is admissible and complete, Standley claims A^* with OD is admissible and complete as well [92]. Optimality of $A^* + OD$ cannot be guaranteed due to the subdivision of the time steps. The subdivision could namely result in a situation in which agent i is considered before agent j , and where agent i is not allowed to move in the position of agent j , while j still has to be considered.

Independence Detection (ID) was introduced by Standley [92] as well. In ID, the full MAPF problem is split into smaller problems for only a few agents. ID first assigns each agent to a group and finds a solution for each group (with A^*) while not considering the other groups. The solution of agents in the different groups is then checked and in case a conflict is found between the plans, the algorithm first tries to find a conflict-free path for one of the groups with the same solution. If this is not found, two groups are merged into one group and planned for together. This continues until no conflicts occur between the solutions found for the different groups. Standley claims that ID is complete when it is used with a complete search algorithm such as A^* or OD. ID was shown to work both with A^* and $A^* + OD$ and tested the different combinations against HCA* (discussed later in this report) by Standley. From the different combinations of A^* , OD and ID, the A^* algorithm extended with both OD and ID was found to be the most efficient in terms of computational time.

Other variants of A^* are Enhanced Partial Expansion A^* (EPEA*) [27] as introduced by Felner et al. and Increasing Cost Tree Search (ICTS) [83] as introduced by Sharon et al.. Before discussing these approaches in detail, the experimental results of these solvers will be analyzed with regard to other approaches that will be discussed in this literature study. This is done based on the DAO brc202d map, which is a map with narrow corridors and is therefore considered most representative for an airport layout. Experiments by Sharon et al. [83], have shown ICTS to outperform $A^*+OD+ID$ in terms of solution rate and success rates. In their later work, Sharon et al. [84] compared EPEA* and ICTS against their Conflict Based Search (CBS). It was shown that, on a brc202d map, ICTS outperformed EPEA* and both CBS and MA-CBS outperformed ICTS (in most cases) and EPEA*. It can therefore be concluded that CBS and MA-CBS are dominant over the solvers discussed in this section. It is therefore chosen not to discuss, OD, ID, ICTS and EPEA* any further for application in this research.

5.2.2. Conflict Based Search (CBS) and variants

An example of an optimal multi-agent path finding algorithm is conflict based search (CBS), introduced in 2015 by Sharon et al. [84]. Many variants of CBS have been proposed based on this original algorithm for various applications. As discussed in chapter 3, an adapter version of the CBS algorithm has also been used in existing research at the ATO group at Delft University of Technology as the basis for routing both aircraft and TaxiBots ([9], [29], [30]). For this reason, and the fact that the algorithm forms the basis of many state-of-the-art approaches or can be used in combination with these, the basic CBS algorithm will be discussed below first. Improved versions of it will be discussed afterwards.

Conflict Based Search (CBS) and Meta-Agent CBS (MA-CBS)

CBS uses the A^* algorithm to plan optimal paths for single agents and then evaluates potential conflicts between these paths. The conflicts are then resolved by introducing constraints. Agents then plan a new optimal path, taking into account potential constraints. This goes on until an optimal, conflict free solution to the MAPF problem has been found. More formally, the CBS algorithm consists of two different stages after initialization, namely the high-level search and the low-level search. These different stages will be described below, in line with the description by Sharon et al. [84].

Step 1: Initialization

Given a graph and the start position and the goal position of all agents, individual paths for all agents are computed using the A^* algorithm without taking into account the paths or intentions of other agents. The result of this first step is a set of single-agent paths which might consist of conflicting paths. This is similar to the initial paths planned in the example problem from Figure 5.2.

Step 2: High-level search

In this second step, all single-agent paths, consisting of a node location for each agent for each time step, are compared. This is shown in the top of the constraint tree in Figure 5.3. Note that this constraint tree contains paths and constraints related to the example MAPF problem in Figure 5.2 from Sharon et al. [84]. If multiple agents are observed to be at the same node at the same time point, constraints are introduced. A constraint

forbids an agent from occupying a node at some time point. This is shown in the bottom of Figure 5.3, where a constraint is introduced for either agent 1 or agent 2, of whom only one is allowed to occupy node D at time point 2.

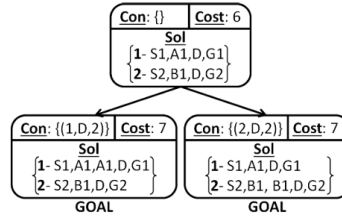


Figure 5.3: Constraint tree used in Conflict Based Search to resolve a conflict at D [84].

Step 3: Low-level search

Taking into account the constraints introduced in the high-level search, a new individual path is planned again for all agents. If the single-agent paths do not contain any nodes that are occupied by multiple agents at the same time, a conflict-free solution to the MAPF is found. If two or more agents are still at the same node at the same point in time, another constraint is added in the high-level search and step 2 and 3 are thus repeated. This is done until a conflict-free solution to the problem is found.

CBS is an optimal and complete MAPF solver, which means it finds a shortest path if there exists one [84]. The approach was tested and compared to other approaches on a brc202d DAO map (which is a map without large open spaces and many bottlenecks and thus most similar to an airport layout). It was shown to have success rate of 100% for up to around 15 agents, which is slightly more than EPEA* which returns around 80% solutions for 15 agents. The benefit of CBS over EPEA* increases when the number of agents increases.

However, a weakness of CBS was identified by Sharon et al. [84]. They found that in cases where the paths of multiple agents have many similarities, CBS needs to solve many conflicts before a solution can be found. In other words, the constraint tree as shown in Figure 5.3 is expanded many times before a conflict-free solution is found. An adapted version of CBS, called meta-agent conflict based search (MA-CBS) was therefore proposed which merges agents that are strongly coupled into a single meta-agent [84]. By merging these agents, the number of conflicts that needs to be resolved by CBS is found to be lower.

Whereas the CBS algorithm only has the option to create new branches in the constraint tree when a conflict occurs, MA-CBS thus has the possibility to merge conflicting agents into one meta-agent. To determine whether a conflict is resolved by branching or merging, MA-CBS uses a conflict-bound parameter B . The conflict-bound parameter B is a threshold for the number of conflicts between a number of agents before they are merged. If the number of conflicts found is $< B$, agents are not merged and if the number of conflicts $> B$, agents are merged. Note that when considering MA-CBS(∞), thus with $B = \infty$, agents are never merged into one meta-agent and the problem is therefore similar to CBS. When considering MA-CBS(0), thus with $B = 0$, agents are merged directly when a conflict occurs so that the problem is similar to ID [84]. MA-CBS was tested on the brc202d DAO map as well with a branching factor of 1, 10 and 100. Performance with $B = 1$ show similar results to EPEA*, with $B = 10$ results are similar to CBS for the first 20 agents but then decrease with respect to CBS and for $B = 100$, the success rate is slightly better than that of CBS. Although the difference between the two is larger (MA-CBS better than CBS) for maps with a lot of open spaces, this difference is thus small on the brc202d map. This last map is considered as most representative for airport layouts and therefore the expected difference between CBS and MA-CBS for application at airport layouts is small.

GCBS, BCBS and ECBS

Enhanced CBS (ECBS), bounded CBS (BCBS) and greedy CBS (GCBS) are three suboptimal variants on the original algorithm and were introduced by Barer et al. [8]. The main motivation for these approaches is that, in CBS, nodes in the constraint tree that are close to the solution but not yet fully optimal are not expanded any further. The aim of these suboptimal algorithms is to trade optimality for a better runtime. Greedy CBS (GCBS) prioritizes nodes in the constraint tree (high-level search) that seem closer to the goal node. This closeness to the goal is estimated using conflict heuristics, described in the original paper. The low-level

search was also adapted by using A* with the conflict heuristic for single agent path finding, which returns a suboptimal path. Although GCBS is complete under conditions, it is unbounded suboptimal, meaning that there is no guarantee that the solution is within a finite factor of the optimal solution. If solution quality is not a requirement GCBS is a suitable way in order to find a solution quickly.

Barer et al. also propose Bounded CBS (BCBS) and Enhanced CBS (ECBS) on the other hand, which use a focal search in order to obtain a bounded suboptimal solution. A bounded suboptimal solution is within a certain factor from the optimal solution. The focal search works with two lists of nodes, an OPEN and a FOCAL list. The OPEN list is similar to the list used in A* and the FOCAL list is a subset of this list. The nodes that are in the OPEN list are determined by functions f_1 and f_2 and the suboptimality factor w . The FOCAL list is filled with nodes from OPEN that satisfy $f_1(n) \leq w \times f_{1_{min}}$, where $f_{1_{min}}$ is the minimum f -value from the OPEN set. f_2 is used to determine which node from FOCAL is expanded. Under the condition that f_1 is admissible, the solution of the focal search is at most w times the optimal solution. In BCBS, the focal search is used both in the high and the low level searches of CBS. The high level is a focal-search(g, h_c) meaning that $f_1 = g(n)$, the cost of the CT node and $f_2 = h_c(n)$, the conflict heuristic (as in GCBS). The low-level is a focal-search($f(n), h_c(n)$) where f_n is the f -score as used in A* and h_c is the conflict heuristic again. The parameters for BCBS are w_H and w_L representing CBS high-level focal search suboptimality factor and low-level search suboptimality factor respectively. The solution of BCBS(w_H, w_L) is suboptimal with a factor $w = w_H \times w_L$ with regard to the optimal solution. The BCBS approach is shown to have a higher success rate than the regular CBS algorithm, while returning a solution that is a bounded factor w from optimal. On a brc202d DAO map and $w = 1.01$ Barer et al. shows that the last point where CBS solves for 100% of instances is for around 15 agents, while this point is around 30 agents for BCBS.

As noted by Barer et. al, the distribution of the suboptimality factor between the high-level search and low-level search is a challenge. As in BCBS, Enhanced Conflict Based Search (ECBS) uses a focal search for the high and low level of CBS. It uses an *OPEN* and *FOCAL* list in the high level and a *OPEN_i* and *FOCAL_i* list in the low level where the i represents the agent affected by the constraint set in the high level. Two values are passed from the low level to the high level in ECBS, $n.cost$ and $LB(n)$ representing the cost of node n and the lower bound of the optimal solution respectively. FOCAL is represented as follows [8] (p.24):

$$FOCAL = \{n \mid n \in OPEN, n.cost \leq LB \cdot w\} \quad (5.1)$$

The cost of all nodes in FOCAL are thus within a factor w of the optimal solution. Like in BCBS, the final solution of ECBS is guaranteed to be within a factor w of the optimal solution. ECBS was compared to CBS by Barer et al. The success rate of ECBS was found to be significantly higher than for CBS, while solution cost is guaranteed to be within a bounded factor from the optimal solution. On the previously discussed brc202d DAO map with $w = 1.01$, ECBS returned a solution for 100% of agents up to around 70 agents, which is much higher than that of CBS (around 15) and BCBS (around 30). Barer et al. appoint the higher success rate of ECBS above BCBS to the higher flexibility of ECBS in the high level search when the low level returns low cost solutions.

From the three variants of CBS as proposed by Barer et al., the solutions found by GCBS are not within a bounded factor from the optimal solution. It is therefore considered as inferior to BCBS and ECBS that were shown to return good solutions for many agents much faster than CBS. Due to the significant higher success rate of these last two algorithms at a cost close to optimality, these approaches could be a suitable candidate for the application in the area of airport ground surface operations.

5.2.3. Priority Based Solvers

The final group of classical MAPF algorithms that will be discussed are priority based solvers that determine priority and then solve for path for individual agents in the priority sequence. In general, concerns with priority based solvers are that their optimality and completeness is dependent on the way priorities are set. However, the priority setting in these algorithms does often allow for manually setting priorities as input for the algorithm. At airports, it might be desirable to give one aircraft priority over another (for example an aircraft that is running late for its Calculated Take-Off Time). For this reason, it is interesting to explore this set of algorithms in more detail.

CA*, HCA*, WHCA*, CO-WHCA* and CO-WHCA*P

Silver proposed three variants of the A* called Cooperative A*, Hierarchical Cooperative A* and Windowed Hierarchical Cooperative A* [88]. The algorithms use a combination of priorities and individual path plannings in order to find a solution to the MAPF problem. Thereby they split the MAPF into a problem of single agent path finding problems. In Cooperative A* agents are first ranked in a priority order. A single agent search (with A*) is then performed for the agent highest in priority and its path is saved in three dimensional reservation table, containing (x, y, t) dimension for x -location, y -location and time step respectively. The path for the higher prioritized agent is reserved in this table after which it should be avoided by agents lower in priority that plan their path afterwards. Single agent searches are performed until all agents have found a path. It is important to note that the CA* is not complete, as agents that are high in priority could block points in the reservation table that are needed for agents lower in priority to find a solution. Due to this greedy behavior of high priority agents, the CA* is also not optimal. Silver proposed an improvement of CA* called Hierarchical Cooperative A* (HCA*), which is essentially the same as CA* but uses the pre-computed abstract heuristic in the single agent A* search. This abstract heuristic represents the exact distance to the goal node and is determined by performing a search from the goal node to the start node. Although HCA* finds solutions faster than CA*, it still uses a randomly set priority order which could lead to incomplete and suboptimal solutions.

Windowed Hierarchical Cooperative A* (WHCA*) [88] was developed by Silver to limit the incompleteness of CA* and HCA*. Besides that, this approach is more efficient than CA* and HCA* as it does not search for a full route in a three dimensional state space. Instead, WHCA* uses a search window w in the cooperative search. Agents only look for a path w steps in the future while still using the previously described abstract heuristic (which is determined by full depth search) so that the search is in the direction of the goal node. Events in time steps after w are ignored up to the point where the time window is shifted forward. This use of a search window makes WHCA* much more efficient than CA* and HCA*. It should be noted however, that WHCA* only limits the incompleteness of HCA* and CA*. It still reserves a path in the reservation table for w time steps in the future, which could still block a feasible path to the goal for another agent. The use of the search window in this algorithm is potentially interesting to explore further as it can be used in online MAPF, where agents are constantly added to the environment. This will be explained further in the trade-off section at the end of this chapter.

Finally, two variants of WHCA* were introduced by Bnaya and Felner [11] called Conflict Oriented WHCA* (CO-WHCA*) and Conflict Oriented WHCA* with prioritization (CO-WHCA*P). Instead of introducing a reservation for every (x, y, t) pair, CO-WHCA* only reserves a time window around a (x, y, t) pair in case a conflict occurs with another agent. A conflict owner is appointed which gets the right to make a reservation in the reservation table, while the other agent is not and thus has to plan around this conflict. In CO-WHCA*P the right to make such a reservation is determined based on the likelihood of finding an alternative path. The solution cost is determined for all possible priority orderings of the agents involved in the conflict. The agent in the top of the ordering with the lowest cost is then chosen as conflict owner. Both CO-WHCA* and CO-WHCA*P were shown to outperform WHCA* in terms of success rate and solution cost on DAO maps [11]. Although CO-WHCA*P found a slightly better solution than CO-WHCA* and WHCA*, its runtime was around 90x slower than CO-WHCA* and around 2.5x slower than WHCA* with differences in solution cost of only around 5%.

Conflict Based Search with priority (CBSw/p) and Priority Based Search (PBS)

Two prioritized variants of CBS were introduced by Ma et al. [52], namely CBS with priority (CBSw/p) and Priority Based Search (PBS). Instead of using an initial priority order as in the prioritized approaches as proposed by Silver [88], these approaches consider different possibilities for the priority orderings of agents. This is done by exploring a state space consisting of different priority orders in the high level search. CBSw/p uses a best-first search to do so while PBS uses a depth-first search. Both approaches will be discussed more elaborately below.

Conflict Based Search with priorities (CBSw/p) works as follows. Just like CBS, CBSw/p splits the search in a high level and a low level search. In the high-level a constraint tree is kept and just like in CBS, the node with the lowest cost in the constraint tree is expanded next. CBSw/p is different from CBS in the sense that it also stores a priority order in each node in the constraint tree. At the start of the search, the set of priorities is empty. If CBSw/p expands a node in the constraint tree, it tries to generate two child nodes for the

agents involved. In one child node, agent i has priority over agent j and in the other the priority is the set the other way around. A child node is only generated if the parent node does not have the reversed priority order already in it. In the generated child node(s), the agent lower in priority is constrained to use an edge or vertex at a certain time point. The priority order is then added in the respective node of the constraint tree and is inherited by child nodes if expanded from this node (just like the plans and sum of cost). A low-level search similar to the low-level search of CBS is done which returns a new set of paths taking into account the constraints. The cost of the node in the constraint tree is updated and as long as there are still conflicting paths the constraint tree is expanded further. This continues until all agents have found a conflict-free path to their goal.

Priority Based Search (PBS) is another prioritized algorithm introduced by Ma et. al [52]. As opposed to CBSw/p, PBS uses a depth-first search in the high level search to explore different priority orderings. It uses a priority tree (PT) in the high level that contains a priority, sum of costs and plans for all agents. Note that PBS does not store constraints in the PT nodes which is the case in the constraint tree used by CBS and CBSw/p in the high level. For a node in the priority tree, in which agent i has priority over agent j , there cannot be any conflicts between these agents. The high level search works as follows. At the start of the search, PBS takes an initial priority order (which can be empty) as input. It then runs the low level search for all individual agents, where agents lower in priority avoid the paths of agents higher in priority. If collisions occur between agents, additional priorities should be set in the high level. PBS resolves the conflict that occurs first by expanding a node in the PT into two generated child nodes. One with agent i priority over agent j and one with agent j priority over agent i . The low level search is then performed again with these added priorities and lower prioritized agents again avoid paths of higher prioritized agents. This is continued until a conflict-free path is found for all agents from their start to their goal.

Both CBSw/p and PBS are unbounded suboptimal. However, Ma et al. claims that CBSw/p often produces solutions that are optimal or near-optimal while being more efficient than CBS. PBS is claimed to be even more efficient and was shown to return near-optimal solutions as well. Ma et al. performed experiments with both of the above algorithms on a DAO brc202d map. The approaches were compared to one another and to CBS and CA* (referred to as FIX in his work). CBSw/p was shown to have better success rates and running times than CBS, but showed a lower success rate and longer runtime than CA*. PBS shows higher success rates than both CBSw/p and CA*, while the runtime was only slightly higher than CA*. On the brc202d DAO map, both CBSw/p and PBS were shown to return solutions that are very close to optimal which was not the case for CA* [52]. Although the algorithms are unbounded suboptimal, they have thus been shown to return solutions that are very close to optimal.

5.3. MAPF for real-world robots

The algorithms for solving the MAPF problem discussed previously all make several assumptions that do often not hold when considering real world robot applications [51]. Main assumptions of most MAPF algorithms are that they assume that ([13], [4]):

- Plan execution is perfect.
- Agents perform one action in exactly one time step.
- Agents occupy only one single location in each time step.
- Time steps are discretized.

However, these assumptions do not always hold in real world application of MAPF. This is also the case for the application in the context of airport ground surface operations. If not accounted for during or after planning, the actual execution of the plan could be far from in line with the output of the MAPF solver. This could negatively affect the safety and efficiency of the execution. Effects of not accounting for these assumptions would therefore be unacceptable when planning for a real world application such as airport ground surface operations. The subsections in this section discuss five main differences between classical MAPF and MAPF in a setting of airport ground surface operations with real robots. Approaches that account for these shortcomings by adaption of previously discussed MAPF solvers will be discussed in this section. [section 5.4](#) will then discuss another perspective on approaching the challenges mentioned below.

5.3.1. Perfect vs. imperfect plan execution

The first assumption of MAPF solvers is that they assume that the execution of the plan is perfect. In reality, there are several reasons why execution cannot be assumed to be perfect. As discussed by Russel and Norvig [64], sources of uncertainty in the real world could be: a delay in adherence to motion commands, friction or slip. Besides that, machinery can break down and, as the airport is part of the real world, other unforeseen disruptions can occur. A common way to deal with these uncertainties is by monitoring the execution of the plan and replanning when a certain condition is satisfied [64]. These aforementioned sources of uncertainty are difficult to foresee and some form of execution monitoring and replanning should be incorporated in the system to work successfully. In the current airport ground surface operation, this task is under the responsibility of both aircraft pilots that observe their environment and ATC that observe the complete traffic situation.

Although the above mentioned sources of uncertainty are unavoidable, uncertainty can also be introduced due to an inaccurate model of an agent and the environment used during planning. Neglecting acceleration and deceleration for aircraft agents for example and planning with the assumption that agents move at maximum velocity, will lead to big differences between the original plan and the execution. One way to deal with these additional uncertainties is to monitor and replan if the deviation from the original plan is too large. However, the downside of these approaches is that replanning is usually time-extensive [34]. Other ways to deal with this second second type of uncertainties is to prevent them by including a more accurate model of the agents. By already taking information about the agent's kinematic properties for example, this second type of uncertainty in the plan execution can be reduced. Note that some level of uncertainty in the plan is unavoidable, even if a more accurate model of the agent and the environment is considered. Approaches discussed in the following sections will discuss ways of incorporating more information about the agent and the environment in the planning phase so that the uncertainty is reduced.

5.3.2. Uniform vs. non-uniform edge traversal times

In classical MAPF, agents are assumed to either wait at a node or move to another node during one time step. When representing in airport in the real world however, the edge lengths are not uniform and therefore actions have a different duration (note that this is still under the assumption of constant velocity). The weights of the graphs can be updated accordingly. In this way, the previously discussed algorithms can be used relatively easily by just changing the weight of an edge into the traversal time instead of 1. The previously discussed MAPF approaches can then still be used with a new graph $G(V, E)$ with vertices V and weighted edges E .

5.3.3. Point agents vs. geometric shaped agents

In classical MAPF, agents are assumed to occupy one single vertex at each time step. At an airport, agents could however occupy multiple vertices and edges when they are formally not at that node but at a neighboring node. This is especially the case in dense areas such as the apron areas. To illustrate this, a example situation from Amsterdam Airport Schiphol is shown in 5.4(a). This is the west side of the G-apron, where there are 3 taxiways that could be represented as a graph as shown in 5.4(b). Taxiways A-B and E-F are wingspan restricted at 36 meters wingspan but can be used simultaneously. Additionally, the middle taxiway C-D is not available if either A-B or E-F is in use and vice versa. Whereas classical MAPF algorithms would only prevent simultaneous movements of aircraft on one of the edges this is not sufficient in this context. Movement between A-B and E-F will namely only be constrained if aircraft actually move between these points and not if another aircraft moves from C to D.

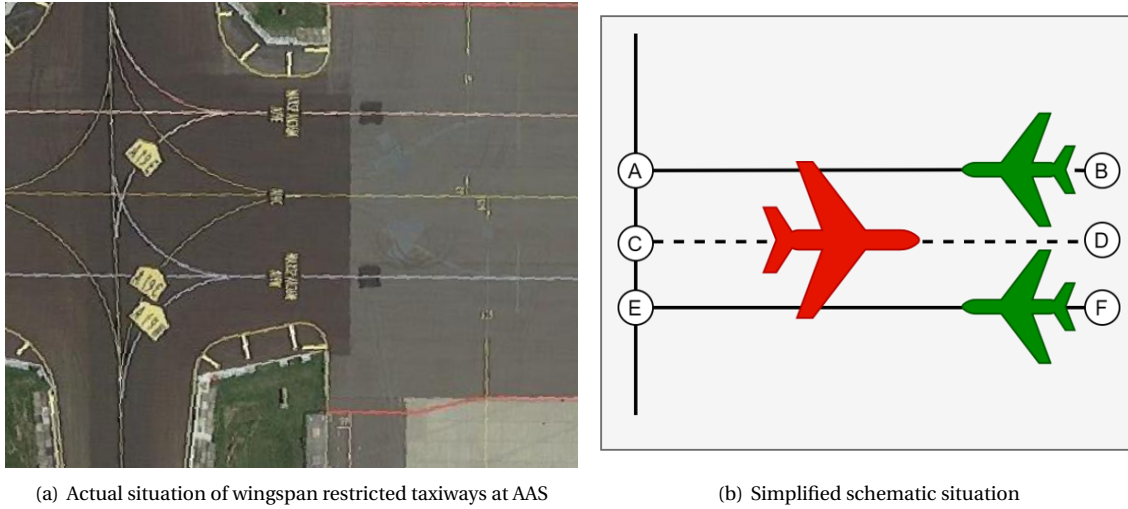


Figure 5.4: Situation where classic MAPF with point agents could result in a collision between the red aircraft and the green aircraft.

The situation as sketched in Figure 5.4 has been identified in literature before on a more general level. Stern et al. [94] refer to Hönig et al. [35] to formalize three new conflicts (next to the ones discussed in Figure 5.1) that could occur when agents have a geometric shape:

- **Vertex-to-vertex conflict:** Occurs for example if the red agent in 5.4(b) is on node/vertex C while another agent occupies A.
- **Edge-to-edge conflict:** Occurs for example if the red agent in 5.4(b) traverses edge C-D while another agent traverses edge A-B.
- **Edge-to-vertex conflict:** Occurs when one agent traverses an edge while another agent is blocking a node.

Although not considered in the classical MAPF approaches as discussed earlier, this shortcoming can be accounted for in several ways. As noted by Li et al. [46], one way is to design the environment with a low resolution, in such a way that classical MAPF algorithms can be applied even with agents with a volume. However, Li et al. also notes that this leads to a decreased performance of the algorithms. Adapted versions of CBS, namely Large Agent CBS (LA-CBS) and Multi-Constraint CBS (MC-CBS) were therefore proposed by Li et al. to account for geometric shapes in path planning and will be discussed below.

LA-CBS and MC-CBS

As classical MAPF solvers assume that each agent only occupies a single point at each time point, Li et al [46] formalized a new problem of multi-agent path finding for large agents (LA-MAPF). In this problem, large agents are defined as agents that could occupy a set of points in a graph at any time point. In LA-MAPF, agents are thus assumed to have a volume and safety distances between agents can be introduced. LA-MAPF is an adapted version of CBS only in the way that it defines conflicts in the high-level. The previously discussed vertex-to-vertex conflict, edge-to-edge conflict and edge-to-vertex conflicts are taken into account in the conflict detection and therefore lead to the generation of additional CT nodes. As discussed by Li et al, constraints for closely related conflicts will be added one by one in this version of CBS so that many additional CT nodes will be generated in the high-level.

To improve the efficiency of the search in the CT tree for this same problem, Multi-Constraint CBS [46] was introduced, capable of adding multiple constraints in one single expansion of the CT in the high-level. In this way, related conflicts can be added in one expansion of the CT instead of multiple as would be the case in LA-MAPF.

Buffers

Another way to account for agent their shapes is to introduce buffers or safety intervals around a node. This is however only a way to account for the size of agents around a single vertex and therefore it does not account

for situations where two agents block the another vertex (vertex-vertex conflict) or edge (edge-edge conflict) when they occupy another one. Note that edge conflicts close to a node around which a buffer is introduced will still be prevented in this way and the approach is thus suitable for environments that are discretized with a low resolution. An example of an algorithm that uses buffers to account for agent their sizes is MAPF-POST [34] which will be discussed later in this section.

5.3.4. Agent Kinematics

Additionally, kinematics cannot be neglected for real world agents. Kinematics are usually ignored in classical MAPF approaches which means that plans could be infeasible to execute ([13], [34]). Aircraft for example, cannot move or stop instantaneously as they need time to accelerate and decelerate. Besides that, aircraft usually taxi through corners at a lower velocity with respect to straight taxiways. Depending on the length of the edges in the airport layout, aircraft might not even be able to accelerate to full velocity on one single edge. Three approaches to account for kinematics will be discussed below.

MAPF-POST

One approach to take kinematics and non-uniform edge lengths into account is presented by Hönig et al. [34] with their MAPF-POST algorithm. MAPF-post uses the output of a complete MAPF solver and then post processes it using a simple temporal network. The result is a more feasible plan execution schedule that takes into account agent's velocity limits and introduces slack to cope with delays in the plan execution and introduces safety distances. The working principle of MAPF will be explained briefly below.

Provided a complete solution from a MAPF solver, the agent paths are converted into an augmented temporal plan graph by adding safety markers that ensure a safety distance between the agents. The augmented temporal plan graph contains all relations and dependencies between the plans of individual agents. This augmented temporal plan graph is then transformed into a simple temporal network with vertices representing events and edges representing constraints between events. The constraints are in the form of a lower bound and upper bound indicating the time period in which one event must be scheduled with regard to another event. These bounds are set based on velocity limits and edge lengths. The lower bound is determined by dividing the edge distance between the locations of 2 connected events by the maximum velocity of an agent. The upper bound is only set to a value unequal to infinity if there is a minimum velocity that has to be considered. The events in the simple temporal network and the edges (representing constraints) can then be used in a graph-based optimization or solved as a linear programming problem. The result is a make-span minimal plan satisfying the simple temporal network constraints. As opposed to the output of the MAPF solver, this plan execution schedule accounts for maximum velocities and buffers for imperfect execution and safety.

MAPF-POST was evaluated both with simulation experiments and in tests with actual robots and was found to successfully post-process the MAPF solution from a MAPF solver into a plan execution schedule. A major downside of this approach is however, that it takes into account the kinematics only after the paths have been determined by the MAPF solver. It is therefore not possible to take the kinematics into account during planning. The effects of kinematics are however expected to impact the plan significantly as aircraft are relatively heavy robots and therefore need time and space to accelerate and decelerate. An approach that takes into account these factors in the initial MAPF solution is therefore preferable for the application in airport ground surface operations. This could make the MAPF-POST approach less suitable than other approaches for application in this research.

Forward Simulation

In previous studies at ATO a so-called forward simulation of the agents their paths was performed in order to include agent their kinematics such as acceleration, deceleration and velocity limits ([9], [29], [30]). First, the Dijkstra algorithm is used to find a shortest path for each agent on a graph with edge lengths equal to the edge distance. Conflict detection and resolving conflicts is then done by localized agents placed at every taxiway intersection. These agents use the edge distance, the agent's acceleration and the agent's maximum velocity in order to determine the aircraft agent their arrival times at the localized ATC agents. Turn angles larger than a predefined maximum angle were simulated at a lower, predefined maximum turn velocity. If multiple agents were predicted to be in conflict, one of the agents would be constrained and rerouted by using the Dijkstra's algorithm to resolve the conflict. Interactions between agents were not accounted for in the forward simulation, which means that the prediction assumes no disruptions along the route.

Note that the output of the forward simulation is a set of **predicted arrival times** at waypoints along the path. The actual execution of the plan can thus still be different in case of disruptions in the system or if other aircraft move over the airport. The resolved conflicts were not guaranteed to actually occur in the execution of the plan. In the same way, conflicts that occur in the execution but were not anticipated remain unresolved. This could lead to unnecessary reroutings or unexpected conflicts which negatively influence the optimality of the solution. Polydrou [59] therefore focuses on the improvement of this forward simulation as part of ongoing research. He uses machine learning techniques in order to predict the taxi times more accurately. Another option to reduce the uncertainty in the forward simulation is to only detect and resolve conflicts that are predicted to happen in a predefined future time window. In general, these methods of forward simulating predefined paths exclude the possibility to take into account the kinematics during path planning already. A path avoiding corners could for example be preferable over a path with many corners due to its shorter traversal time (although the distance of the first path could be shorter).

Other approaches

Methods from the area of Multi Robot Path Planning on the other hand, can take into account the feasibility of an agent's (robot) motion between two points during planning already efficiently. In this way, kinematics can be considered during planning already and the initial path planned for the robot therefore contains more accurate arrival times at waypoints along the route. Ideas from this area will therefore be discussed later in this report. As noted before in this section, uncertainty is not eliminated from the system by doing this as, for example, disruptions can still occur.

5.3.5. Discretized time vs. Continuous time

The discrete time step assumption of many MAPF approaches does not hold in real life because time is continuous. With uncertainty in the plan execution or with move actions with arbitrary duration, it is not enough to set constraints at only one single time point. Atzmon et al. [6] therefore introduced the idea of defining conflict intervals as a range between two time steps during which a specific agent is not allowed to occupy a certain node. The idea of conflict intervals was introduced by Atzmon et al. in order to develop more robust plans, but can be used wider to account for planning in continuous time. Although this approach does enable reasoning about constraints with continuous time, the determination of action durations is still challenging with MAPF approaches that use discrete time reasoning.

Stern [93] argues that approaches that reason with continuous time might be inefficient in determining the duration of waiting actions. When considering gates, aprons and holding operations at the airport it is expected that waiting actions will be part of the plans. Safe Interval Path Planning (SIPP) [58] is most commonly used to deal with this problem, by using safe intervals and conflict intervals. This approach can be used successfully to reason efficiently about situations in continuous time. As will be discussed elaborately in the next section, examples of approaches that use versions SIPP are found in the domain of motion planning for robots (e.g. Continuous CBS (CCBS) [4]).

5.4. Motion-Based MAPF Approaches

Another family of approaches is from the area of multi robot path planning or motion planning. These approaches are grouped into one as they all use a different state representation and mapping between states in the state-space. Instead of defining states as (location, time) pairs, these approaches define states as (configuration, interval) pairs. This makes the search more efficient. Assumptions made in the algorithms below are compared to the assumptions made by classical MAPF approaches:

- Plan execution is perfect (Same as in MAPF)
- Action durations can be longer than one time step (MAPF: Agents perform one action in exactly one time step)
- Agents can occupy multiple locations (MAPF: Agents occupy only one single location in each time step)
- Time is continuous (MAPF: Time steps are discretized)

In this section, first SIPP will be introduced as a single-agent path finding algorithm that forms the basis for this different form of reasoning. Then, other approaches that extend SIPP for multi-agent path finding will be discussed more elaborately.

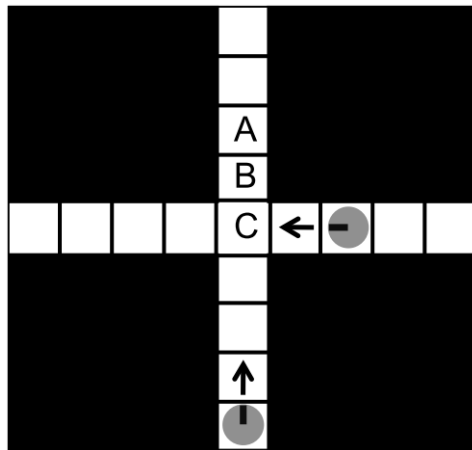
5.4.1. Single Agent Path Finding with Safe Interval Path Planning (SIPP)

Before moving to the evaluation of multi-agent motion planners, the Safe Interval Path Planning (SIPP) [58] algorithm will be discussed that forms a basis for the motion planners considered in this section. Then a variant of SIPP, SIPPwRT [53] is discussed that takes into account the sizes of agents. Finally, a w -bounded suboptimal variant of SIPP, SCIPP [13] is discussed that also reasons about collisions efficiently and that can be used together with ECBS-CT as discussed later in this section.

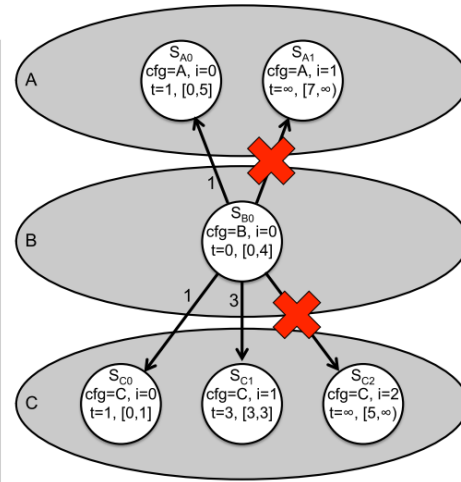
SIPP

The Safe Interval Path Planning (SIPP) algorithm [58] is a decoupled approach that takes a predefined priority setting and then plans paths for single agents based on this priority using a variant on the A* algorithm. The main difference with previously discussed decoupled solvers such as HCA* is that SIPP is the way in which states in the state-space are defined. SIPP searches in a state space that consists of configuration and interval pairs rather than location and time step pairs thereby reducing the search space. A safe interval is defined as an interval without any conflicts and therefore usually contains multiple time points. For a search space consisting of many time steps the search space is therefore smaller in SIPP resulting in a faster approach than planners that use a time dimension [58]. SIPP can be used to reason about continuous time and motions with a non-uniform duration and is therefore suitable for application in continuous time and for different action durations.

A configuration is defined as a combination of the agent's physical location and additional information about the state of the robot such as velocity or heading. SIPP thus defines a state s as a configuration interval pair. $g(s)$, $h(s)$, $parent(s)$ and $time(s)$ are saved for each state s . They represent the g -score (as in A*), the heuristic (as in A*), the parent state of s and the earliest possible arrival time of the agent in cfg respectively. SIPP works similar to A*, as it chooses to remove the state with the lowest f -value from OPEN and generates successors for this state. The way in which successors are generated is different from A*. SIPP considers the set of motions m , $M(s)$, that are possible from state s . The resulting configuration from this action is checked for its safe intervals. The earliest arrival time in each of these intervals is then determined and checked if it is collision free. If a feasible motion is found to a new state s' (consisting of a configuration and a safe time interval), this new state is added to the successors. Successors are then considered in the same way as in A* (expand the node with the lowest f -score next). An example of the generation of successors in the SIPP algorithm will be given below.



(a) Example problem with three configurations (A,B,C) and an agent starting at B with two moving obstacles to avoid.



(b) Expansion of the start state at B, S_{B0}

Figure 5.5: Example of successor generation in SIPP [58].

Example SIPP problem (based on Phillips and Likhachev [58])

For this example the situation as shown in 5.5(a) is considered. Three configurations are shown A, B and C. The agent that is planned for with SIPP is located at B initially. Two dynamic obstacles move through the environment in the shown direction and should be avoided by the agent starting from B.

Step 1: In the initialization of SIPP, the safe intervals are determined for each configuration. Under the assumption that the moving obstacles move one cell per time step, the safe intervals for A are $[0, 5]$ and $[7, \infty)$. At $t = 6$ this cell is namely occupied by the upward moving obstacle. The safe intervals for C are $[0, 1]$, $[3, 3]$ and $[5, \infty)$ as the agent moving from left to right occupies it at $t = 2$ and the obstacle moving upward occupies it at $t = 4$. The safe interval for B is from $[0, 4]$ which means the robot can at maximum wait till time step 4 in this cell.

Step 2: States are expanded from the start state S_{B0} as shown in 5.5(b). Potential successors locations are the neighbors of B, like in the A* algorithm. There are different ways how to arrive in these successors. For A, the robot can move immediately and arrive the earliest at $t = 1$. This is within the first safe interval of $cfgA$ and therefore the move to state S_{A0} is feasible. If the robot would arrive in the second safe interval at A, it would have to wait till time step 6 and then move to A to arrive at $t = 7$. Waiting in B is not possible longer than time step 4 as the safe interval at B ends then. The expansion of S_{A1} is therefore not a valid option. There are two valid options to go to from B to C. Moving directly to C would result in state S_{C0} , arrival at $t = 1$ the first safe interval. Waiting at B for 2 time steps and then moving to C results in state S_{C1} . Note that arrival in the third safe interval of C is not possible, as there is no feasible motion from B to C after waiting at B for 4 time steps. A collision with the upward moving agent would be unavoidable in this case.

Step 3: The resulting successors from start state S_{B0} are thus S_{A0} , S_{C0} and S_{C1} . The g -score for these new states is updated with the time needed to move from the start state to this state. Also the f -score is updated by adding the g -score and the heuristic. The state is then added to the OPEN list after which the state with the lowest associated f -score is expanded next till the goal is reached.

The use of intervals makes SIPP suitable for efficient application in continuous time [13]. It can also handle waiting actions in continuous time efficiently, based on the interval start and end times. Besides that, the way in which SIPP determines its neighbors and defines states is based on feasible motions and the resulting configurations. This brings the option to include agent kinematics such as acceleration and deceleration into account during planning and therefore contributes to the interest to explore this algorithm and its variants further.

SIPP with reservation table (SIPPwRT)

Safe Interval Path Planning with reservation table (SIPPwRT) was introduced by Ma et al. [53] as a generalization of SIPP in continuous time which also works for agents with a volume. An initial priority order should be determined in the high-level as input for the algorithm. Safe intervals are then determined per node in the same way as in SIPP. Ma et al. suggest to use a reservation table to store these safe intervals per cell, ordered in ascending order based on their lower bounds. The authors claim this to be an efficient way to keep track of safe intervals and add intervals in case a new path is planned or delete intervals if they are not relevant anymore (in case they end before the current time). If agents have a volume, planning with these safe intervals does not yet guarantee that two or more agents do not collide. Multiple agents that are about to arrive at or leave from a node could still collide. Ma et. al therefore update the safe intervals with time offsets based on the agents volume (or safety radius), configuration and velocity. This works as follows: SIPPwRT uses a safety radius R_i for each agent i representing them as a circular object with radius R_i . Note that increasing this safety bound is similar to introducing a safety margin around an agent. The time offset ΔT represents the time between the time one agent needs to vacate a cell and the time that another agent arrives at the cell. The distance between two agents is D and should be equal to or larger than $R_1 + R_2$ in order for two agents 1 and 2 not to collide. The time offset that should be added to the safe interval is distinguished for three different cases as shown below, based on the description by Ma et. al. A more elaborate description of the calculation of these offsets can be found in Ma et al. [53].

- Agents 1 and 2 move in the same direction with velocities v_1 and v_2 respectively :

$$\Delta T = \frac{R_1 + R_2}{\min(v_1, v_2)} \quad (5.2)$$

- Agents 1 and 2 move in orthogonal direction with velocities v_1 and v_2 respectively :

$$\Delta T = \frac{\sqrt{v_1^2 + v_2^2} (R_1 + R_2)}{v_1 \times v_2} \quad (5.3)$$

- Agents 1 and 2 move in opposite directions with velocities v_1 and v_2 respectively. L is the distance between two cells:

$$\Delta T = \frac{L}{v_1} + \frac{L}{v_2} \quad (5.4)$$

With these time offsets ΔT the time intervals off the configuration are updated so that agents cannot collide. The path found by SIPPwRT is proven to be guaranteed to have no collisions with the paths of other agents [53]. Furthermore, the single-agent path in SIPPwRT is optimal given a certain priority order just like in SIPP. Ma et. al claim the approach to be complete for well-formed multi agent pickup-and-delivery task in combination with Token Passing (TP-SIPPwRT). Unfortunately, the completeness of SIPPwRT has not yet been evaluated for MAPF, but it is expected to behave similarly as SIPP as the approaches do not differ that much. Ma et al. have proposed to generalize existing MAPF algorithms with SIPPwRT so that continuous movements and velocities can be accounted for and to include acceleration and deceleration in SIPPwRT. These two suggestions are especially interesting for the use in an airport environment when dealing with agents with kinematic properties.

Soft Conflict SIPP (SCIPP)

Soft Conflict Safe Interval Path Planning (SCIPP) was introduced by Cohen et. al [13] as part of the low-level solver of the multi agent motion planning technique ECBS-CT that will be discussed later in this section. This low-level solver is a single agent path finding algorithm that is suitable to be used in a FOCAL search and will be introduced in more detail below.

The input for the SCIPP algorithm is a start and goal state of an agent, a suboptimality factor w for the focal search, a set of hard constraints for each cell c and a reservation table from a high level node containing the paths of previously planned agents. The hard constraints are in the form of a cell, time step pair (c, t) and are set in the high-level search. The reservation table contains so-called soft conflicts in the form of a cell, time interval pair $(c, [lb, ub])$ and represent the cells swept during a motion by higher priority agents. In order to choose which node to expand, SCIPP uses the node with the lowest f -value from the FOCAL list. Remember from ECBS that the f -value in the FOCAL list of the low level search is determined by the conflict heuristic h_c . In SCIPP, the conflict heuristic h_c of a node in FOCAL is the number of soft conflicts. This is determined by counting the number of cells in which an agent is in conflict with another agent's path in the reservation table when it moves from the current node to the next node (notice that nodes represent states here). SCIPP maps the soft conflicts and the hard constraints to disjoint safe time intervals. The approach can be considered as an adapted version of SIPP that is suitable for the low level focal search in ECBS-CT, as it is able to account for the conflict heuristic used [13]. This ECBS-CT variant will be discussed more elaborately later in this section.

From Single Agent SIPP to Multi Agent SIPP

The previously described approaches are all single agent path finding approaches. Several approaches have been suggested to use these variants of SIPP in multi-agent planners. In the next section, Enhanced Conflict Based Search Continuous Time (ECBS-CT) [13], Continuous-Time Conflict Based Search (CCBS) [4] and Continuous-Time Conflict Based Search with kinematic constraints (CCBS-kc) [3] will be discussed as approaches that use SIPP successfully for motion planning.

5.4.2. Enhanced Conflict Based Search Continuous Time (ECBS-CT)

Conflict Based Search Continuous Time (ECBS-CT) was introduced by Cohen et al. [13] as a bounded sub-optimal version of E-CBS (discussed earlier). They also introduced Soft Conflict SIPP that was discussed previously in this report. The authors were motivated by the challenge to overcome the theoretical assumptions in MAPF approaches while still using the principles from state-of-the art MAPF solvers.

The problem solved by this algorithm is described as the Multi Agent Motion Planning problem (MAMP), which uses states instead of locations to define the search space [13]. A state in the state space is represented as a discretization of the agent's location, orientation, velocity and potentially other state parameters [13].

An edge between two states is a kinodynamically feasible motion that can have a non-uniform arbitrary duration. The solution to the MAMP problem is a set of connecting motions from the start state to the goal state of an agent. Agents with any shape are considered in this problem in the form of occupying a set of cells corresponding to its shape. The problem is formally described the following, directly obtained from [13]:

”Given an environment represented by a list of cells C , we are given agent $1, \dots, k$, each with an associated graph $G^j = (V^j, E^j)$ and start and goal vertices $s^j, g^j \in V^j$. Each vertex $s \in V^j$ represents a **state** and is associated with a list of cells $\{c_1^s, \dots, c_{m(s)}^s\} \subseteq C$ occupied by the agent while at s . Each edge $e \in E^j$ represents a **motion** and has an associated weight, $w(e) > 0$, that represents its duration. e is also associated with a multiset of cells $\{c_1^e, \dots, c_{m(e)}^e\} \subseteq C$. Each cell c_i^e is associated with a time interval $[lb_i^e, ub_i^e]$ during which it is swept by e after the beginning of its execution.” - (Cohen et. al [13], p. 46)

A plan for an agent consists of a set of motions e each with an associated start time. If two agents are planned to sweep the same cell or occupy it at a vertex at the same time these agents are in conflict. A solution to the above described problem is a combination of plans without any collisions between agents that follow these. With this problem definition, the ECBS continuous time (ECBS-CT) algorithm as proposed by Cohen et al. can be described. The input of the algorithm is a MAMP instance and a suboptimality factor w . Nodes are placed in the FOCAL list in a similar fashion as in the original ECBS algorithm and this will therefore not be described again in this section. ECBS-CT detects conflicts between two agents in the form of a cell and time interval pair. The earliest conflict is resolved in ECBS-CT, which is obtained from the reservation table from the low level search with SCIP. To ensure the suboptimality of ECBS, Cohen et al. translates this constraint into a constraint with a cell, time point (c, t) pair for the high-level search. This constraint is then used to expand the two successor nodes in the high level search. The low-level search is then performed for the two agents that were constrained, taking into account these constraints. The low level search uses a reservation table with time intervals that specify constraints again. Cohen et al. suggest to use an interval map [14] as a mean to efficiently keep track of the constraint intervals in the reservation table. As previously described, the high-level of ECBS-CT needs the earliest conflict from this reservation table which can, according to Cohen et al., be efficiently done by using this interval map. After performing the low-level search, the reservation tables are updated and the new child nodes are placed in the OPEN list and, if they satisfy the conditions to be placed in FOCAL also in the FOCAL list (of the high-level search).

To test the algorithm, Cohen et. al used 2 DAO maps, the arena and the Den420d map. Besides that, two sets of motion primitives is used that describe the possible motions possible from one state to another. One set allows for only 5 motions while the other set allows for 13 motions. Note that the choice of the number of motion primitives determines the branching factor for the node expansion in the low-level search. A high number of motion primitives will therefore lead to a longer running times. On the other hand, the success rate on a dense map was found to be higher for the set with 13 motion primitives as this allowed for more flexibility. In general ECBS-CT was found to be a suitable algorithm to find a solution for realistic MAMP problems [13]. Some remarks should be made regarding the difference between the focus of Cohen et al. and the focus of this research.

Firstly, the feasibility of motions considered in this approach are considered based on the heading angle of the agents. For an airport, the taxiways that an vehicle can use are predefined based on the lines on the taxiways. This means that there is no need for an approach that includes heading angles in the state configuration. However, most turns at an airport are speed restricted and aircraft are restricted by kinematic constraints such as speed, acceleration and deceleration limits. For the scope of this research, it would therefore be more interesting to look at the feasibility of the motion from an acceleration, deceleration and speed perspective instead of checking the feasibility based on the heading angle. This approach of evaluating motion primitives with velocity considerations was also proposed by Cohen et al. in their paper about ECBS-CT [13], which adds to the interest to explore this further.

Secondly, the implementation in ECBS-CT considered swapping of intermediate cells during a motion from one configuration to another. The airport environment is based on taxiways and intersections between those. Besides that, it is relatively easy to define rules for the occupation of an segment in case another segment is occupied. Comparing this to the cell-based environment in ECBS-CT, it is expected that the size of the reservation table for an airport environment will be much smaller and will therefore make the algorithm run faster.

With the above remarks and proposed adaptations, the ECBS-CT approach as introduced by Cohen et al. is expected to be a suitable candidate for further consideration. It is expected that the implementation of acceleration, deceleration and velocity does not harm the working principle of the approach, as it is still based on feasible motion primitives and state representations in the form of a (configuration, interval) pair. The second proposed change, stepping away from the cell environment and moving to a more airport like environment, is expected to make the ECBS-CT algorithm more efficient without harming its functionality. This makes the algorithm a suitable candidate for further consideration in this research.

5.4.3. Continuous-Time Conflict Based Search (CCBS)

Continuous-time CBS was proposed by Andreychuk et al. [4] as a combination of a modification of the original CBS algorithm for the high-level search and a modification of Safe Interval Path Planning (SIPP) for the low-level single agent search. The main differences between CBS and CCBS are that CCBS takes into account the size of agents. It uses constraints over actions and time ranges instead of location and time step pairs and the low-level search of CCBS uses adapted SIPP. The problem solved by CCBS is similar to the previously described MAMP problem, although Andreychuk et al. refer to actions to map from one state to another rather than motions.

To form CCBS, CBS was adapted in order to allow for non-uniform durations and in order to account for conflicts between agents when they move on different edges (an edge-to-edge conflict as described earlier). Besides that, actions in CCBS can start at any moment in time. A collision in CCBS is therefore defined by two actions (a_i, a_j) and start points of these actions (t_i, t_j) , together (a_i, t_i, a_j, t_j) . A conflict is defined as $(i, a_i, [t_{\text{begin}}, t_{\text{end}}])$ where i is an agent, a_i is an action and $[t_{\text{begin}}, t_{\text{end}}]$ is the interval in which agent i is not allowed to do a_i . In the high-level search of CCBS, an unsafe interval of an action with respect to the other agent's action is computed for both involved agents and is defined as follows:

”The unsafe interval of action a_i with respect to action a_j is the maximal time interval starting from t_i in which performing a_i will create a collision with performing a_j at time t_j .” - (Andreychuk et. al [4], p. 41)

This leads to the high-level constraints in the form $(i, a_i, [t_i, t_i^u])$ and $(j, a_j, [t_j, t_j^u])$ for agent i and j and their unsafe intervals with respect to the other agent's action. CCBS then generates two child nodes (as in CBS) of which one will constrain agent i to perform action a_i during its unsafe interval and the other child node will constrain agent j to perform action a_j during its unsafe interval. The high-level search of CCBS is a similar best-first search as in the original CBS approach and will therefore not be discussed any further in this section.

The low-level search however, is different from the low-level search in CBS as an adapted version of SIPP is used. There are two options following from the high-level constraints $(i, a_i, [t_i, t_i^u])$ and $(j, a_j, [t_j, t_j^u])$:

- a_i is a move action with source node A and target node B: If agent i arrives at A at a time step t in the unsafe interval $[t_i, t_i^u]$, the action associated with a move of i from A to B at time step t is deleted. An action that makes agent i wait at A until time t_i^u and then moves to B is added instead.
- a_i is a wait action at node A: The safe interval at A is split up in two safe intervals in order to constrain agent i from waiting at A during the interval $[t_i, t_i^u]$. One of the new safe intervals ends at t_i while the other starts at t_i .

Although the proof is not shown in the paper, Andreychuk et al. claim this adapted SIPP to be optimal given the constraints from the high-level. Besides that, Andreychuk et al. proof that the version of CCBS as presented in his paper is complete and optimality is guaranteed. Geometric conflict detection, between two agents with a volume, can be challenging in this algorithm as mentioned by the author of the paper. As discussed previously it is relatively easy to use rules to define what actions on physical edges and nodes (taxiways and intersections) at the airport constrain the use of other edges and nodes. The challenge of conflict detection with CCBS is therefore considered to be less relevant for the application of this approach in the context of airport ground surface operation. As noted by Andreychuk [3] himself, CCBS does not take kinematic constraints of real world robots into account. A new approach, CCBS-kc, that takes into account this issue was introduced and will be discussed next.

5.4.4. Continuous-Time Conflict Based Search with Kinematic Constraints (CCBS-kc)

As an extension of CCBS, Andreychuk proposed Continuous-Time Conflict Based Search With Kinematic Constraints [3]. It uses the same high-level planner as in CCBS with a slightly adapted low level SIPP planner that prefers states that are dominant over other states. This adapted solver will be discussed below.

The low level search is similar in the basis as it has the same two options to incorporate a high level constraint in the low level (either introduce a wait action or split the safe interval). When incorporating the rotations of robots (or velocities), the search space under consideration can become large due to the high number of possible motions from one state to another. However, Andreychuk notices that it is not necessary to expand states with all possible headings and therefore adapts the SIPP algorithm accordingly. In CCBS-kc he considers robots that can rotate at their place and evaluates motions based on the heading angle. Consider two states s_1 and s_2 associated with the same location but another heading angle. If the g -score of $(g(s_1) + \text{rotation cost to end in heading angle of state } s_2)$ is lower than or equal to the g -score of $g(s_2)$, then Andreychuk describes that state s_1 is dominant over s_2 . All states that can be generated from s_2 can namely also be generated from s_1 at a lower or equal cost. This is referred to state dominance and is implemented as a change in the low-level SIPP algorithm.

This observation can be generalized to velocity motion primitives between two states that take into account acceleration and deceleration limits. Given an initial velocity in the start state, it is always preferable to move to the successor state by trying to accelerate to maximum velocity or to continue at maximum velocity if already reached. The cost of motions that try to move slower will always lead to a lower g -score of the resulting state and therefore do not have to be considered if a faster motion can be performed.

Andreychuk compared CCBS-kc to CCBS. Although both return an optimal solution to the problem, the cost of CCBS-kc solution can be higher than the solution from CCBS, as the kinematics of the agents are considered (only heading angle and time to rotate in the experiments). The algorithms were tested on a empty 16x16 grid, with a run time restriction of 30 seconds for one single agent search. If no solution was found within this time, the algorithm stopped and counted no solution returned. Both CCBS and CCBS-kc were shown to have dropping success rates to lower than 100% already from around 9 agents already. Unfortunately, neither CCBS and CCBS-CT were compared to ECBS-CT. It is expected that results will be similar to a comparison between CBS and ECBS, where ECBS sacrifices optimality for reduced runtime. Following the same line of reasoning, it is therefore expected that the ECBS-CT approach is faster than the CCBS-CT algorithm as it allows for a bounded suboptimal solution rather than a optimal solution that is guaranteed in CCBS-CT. This expectation is supported by the long running times of CCBS and CCBS-CT in the experiments performed by Andreychuk [3].

Although successful in finding optimal solutions that account for agent kinematics and volume, the relatively low success rate of CCBS and CCBS-kc is a major disadvantage of these approaches. Their suitability for successful implementation in the area of airport ground surface operations is questionable for this reason.

5.5. Evaluation of Different Approaches

In this final section, the different approaches that were discussed in the previous sections will be compared. First the classical MAPF algorithms will be compared. Then after, evaluating their general applicability in a context of airport ground surface operations, motion-based approaches will be compared with one another. This section concludes with a proposal to test different algorithms in the context of airport ground surface operations. The algorithms that were considered in this literature study can be grouped as follows (note that CBSw/p and PBS will be compared both in the CBS group and in the priority based group):

- **A*-extensions:** A*, OD, ID, ICTS, EPEA*.
- **CBS and variants:** CBS, MA-CBS, BCBS, ECBS, CBSw/p, PBS.
- **Priority based:** CA*, HCA*, WHCA*, CO-WHCA*, CO-WHCA*P, CBSw/p, PBS.
- **Motion based:** SIPPwRT, ECBS-CT, CCBS, CCBS-kc.

Before moving to the evaluation of these approaches, the evaluation criteria based on performance, applicability and implementation will be elaborated.

5.5.1. Evaluation Criteria

Three categories of evaluation criteria were used to evaluate the algorithm. Performance, applicability and implementation. These will be discussed in more detail below.

- **Performance:** Performance of the algorithm is based on the success rate, runtime and the solution cost. An algorithm with a higher success rate than another algorithm is able to plan paths for more agents than an algorithm with a low success rate. Naturally, algorithms that have a lower runtime will get a higher score as they return a solution faster. The solution cost is evaluated based on the optimal solution. Algorithms that return an optimal solution will get a higher score, bounded suboptimal solvers a lower score and unbounded suboptimal solvers will get the lowest possible score.
- **Applicability:** Applicability refers to how well the algorithm is suited for application in the area of airport ground surface operation. The approaches will be evaluated based on their ability to take into account kinematics, the size of agents and their ability to handle non-uniform action durations and reason with continuous time. Approaches that take these factors into account efficiently during planning score high while approaches that require a lot of assumptions score lower.
- **Implementation:** Finally, algorithms are evaluated on the how complex they are to implement and to what extend previous work is available. Approaches that are hard to implement get a low score while approaches that are relatively easy to implement get a higher score.

The scoring range is from 0 (low) to 3 (high) and equal weights are used for all criteria. In some cases a + sign is added to indicate that a score is in between two categories.

5.5.2. Trade-off

In this section, a trade-off between the different algorithms will be made based on the evaluation criteria as specified above.

A* extensions

The group of A* extensions contains A* (space-time), OD, ID, ICTS and EPEA*. This group will not be further compared in the overall approach, due to their general inferiority to algorithms from the other group on the brc202d DAO map. As discussed earlier, OD and ID were shown to outperform basic A*. Experiments in [83] on the brc202d map have shown that ICTS outperforms A*, OD, and OD-ID in terms of runtime and success rate. Experiments in [84] on the brc202d map showed that CBS outperformed ICTS (in most cases) and EPEA*. For this reason, and the fact that basic CBS will be outperformed by many other algorithms discussed later, the A* extensions are not considered any further for application in an airport ground surface operation setting.

CBS and Variants and Priority Based Approaches

CBS based approaches were compared by several researchers on, among others, the brc202d DAO map. As discussed previously this map is considered to be the most representative for an airport layout due to the relatively low share of open spaces. CBS and MA-CBS were compared with each other in [84], where the MA-CBS performed slightly better than the regular CBS variant in terms of success rate and runtime. The suboptimal versions of CBS, BCBS and ECBS, performed better than CBS on both success rate and runtime. Although their solutions are not optimal, they are within a constant factor w from the optimal solution. For these reasons, they scale better than the optimal variants of CBS. Approaches that are not optimal are PBS and CBSw/p, which were tested on the brc202d DAO map in [52]. Their relatively low score on optimality is mainly because no guarantees can be given on the suboptimality of the solution, which was the case for BCBS and ECBS. However, both PBS and CBSw/p have been shown to return solutions that are 'good' and not far from optimal. In terms of complexity, CBS and PBS score slightly better than the other approaches because they have been considered in previous MSc thesis. However, the difference in complexity of other approaches is not expected to be a lot. Results of the comparison between the different approaches is shown in Table 5.1. As can be concluded from the table, ECBS and PBS are considered to be the best option from this set of approaches.

Table 5.1: Comparison of CBS algorithms based on the brc202d DAO map. Minimum score is 0, maximum score is 3.

	CBS	MA-CBS	BCBS	ECBS	CBSw/p	PBS
Performance						
Success rate	1	2	2	3	2	3
Runtime	1	2	3	3	2	3
Solution cost	3	3	2	2+	1	1
Scalability	1	1	2	3	2	3
Applicability						
Large agents	0	0	0	0	0	0
Kinematics	0	0	0	0	0	0
Continuous time	0	0	0	0	0	0
Implementation						
Complexity	3	2	2	2	2	3
Availability	3	2	2	2	2	3
TOTAL	12	12	13	15+	11	16

From the first group of algorithms, compared in Table 5.1, PBS scored best. It mainly outperformed the other approaches in this group in terms of runtime and scalability. Besides that, PBS has the advantage that initial priorities can be taken into account, which makes it possible to specify general priority rules at the airport (for example: TaxiBots are always lower in priority than aircraft). It should be noted that, although shown to return solutions that are close to optimal, PBS is theoretically unbounded suboptimal. The optimality of PBS should be tested in the application on airport ground surface operations in order to evaluate the quality of the solution in terms of optimality. If solution quality is found to be a problem in this application, ECBS can be a good alternative. It was shown to score slightly lower than PBS mainly because of lower scores on complexity and availability. ECBS is a bounded suboptimal solver, which means that it does return a solution that is of a guaranteed quality (as opposed to PBS).

Another group of algorithms was unfortunately not tested on the same brc202d map, namely the prioritized A* based approaches. Therefore a separate comparison was made between these algorithms as shown in Table 5.2. CA*, HCA* and WHCA* were compared in [88]. WHCA* was shown to outperform both CA* and HCA* in terms of solution rate and runtime due to the limited search window used in this approach. The solution cost of WHCA* was however found to be higher for this reason. Conflict oriented versions of these approaches were discussed and tested in [11], namely CO-WHCA* and CO-WHCA*P. Both approaches outperformed WHCA* in terms of success rate and solution cost. The runtime of CO-WHCA*P however, was a lot longer than CO-WHCA* and slightly longer than WHCA*. Results from this table cannot be directly compared to the results from Table 5.1, as no experiments were done on the same map. From Table 5.1 only PBS and CBSw/p were compared to the algorithms from Table 5.2 in [52]. CBSw/p was shown to have better success rates and running times than CBS, but showed a lower success rate and longer runtime than CA*. PBS shows higher success rates than both CBSw/p and CA*, while the runtime was only slightly higher than CA*. On the brc202d DAO map, both CBSw/p and PBS were shown to return solutions that are very close to optimal which was not the case for CA* [52].

Table 5.2: Comparison of A* Based Prioritized Algorithms. Minimum score is 0, maximum score is 3.

	CA*	HCA*	WHCA*	CO-WHCA*	CO-WHCA*P
Performance					
Success rate	1	1	2	3	3+
Runtime	1	1+	2	3	2
Solution cost	2	2	1	2	2
Scalability	1	1	2	3	2
Applicability					
Large agents	0	0	0	0	0
Kinematics	0	0	0	0	0
Continuous time	0	0	0	0	0
Implementation					
Complexity	2	2	2	2	2
Availability	1	1	1	1	1
TOTAL	8	8+	10	14	12+

From the second group of algorithms, compared in Table 5.2, CO-WHCA* scored best. It outperforms CA*, HCA* and WHCA* and CO-WHCA*P in terms of success rate, runtime and scalability. Although this approach will not be chosen in the final evaluation, it has an interesting property which is the search window. This window, w used in WHCA*, CO-WHCA* and CO-WHCA*P is suitable for handling online MAPF problems, as it used to only look for conflicts a few timesteps ahead. To handle new flights that are added to the planning at an airport effectively, this search window might become useful.

Applicability of CBS and Variants and Priority Based Approaches

The above algorithms have not received a score for applicability. The reason for this is that they are not directly applicable to be used in the area of airport ground surface operations. This is due to the assumptions of these classical MAPF approaches.

A general conclusion that can be made about the MAPF algorithms, both the CBS based and the Priority Based approaches, is thus that they make assumptions that are too simplistic to use them directly in the context of airport ground surface operations. As discussed in section 5.3, there are ways to adapt for the shortcomings of these algorithms. Updating edges to take into account non-uniform traversal times is relatively easy when a single graph is used. Shapes of agents can also be accounted for relatively easy by introducing additional constraints, as shown in LA-CBS and MA-CBS. Restrictions should then clearly be set on edges and nodes and edges that should be blocked in case a vehicle traverses another edge or vertex should be defined manually.

Other shortcomings of traditional MAPF approaches are more difficult to overcome by adapting them. The first is the assumption that kinematics are ignored. MAPF-POST was discussed as a method to account for agent kinematics after planning, by modifying the output of a MAPF solver. However, this approach does not allow to take the kinematics into account during planning already. Performing a forward simulation of the plan after planning suffers from the same disadvantage. The second shortcoming of traditional MAPF that is more difficult to overcome is its planning with discretized time. Although the use of conflict intervals have been proposed, determining arbitrary actions durations is still challenging. The number of states increases significantly if different kinematic values are considered in the single agent searches. Motion based approaches, based on the SIPP low-level solver, do not have these disadvantages and will be discussed next.

Motion Based Approaches

A group of approaches that solves these shortcomings of traditional MAPF approaches is the group of motion-based approaches based on the single agent path finding algorithm Safe Interval Path Planning (SIPP). Instead of defining states as (location, time) pairs, these approaches define states as (configuration, interval) pairs. The configuration, as part of a state, contains all the information that defines the state such as velocity or heading angle. This approach has the advantage over the more classical MAPF solvers that they can be used to reason about continuous time efficiently. Also kinematics can be included in the planning efficiently. Before being expanded in the search space, the feasibility of the motion can be checked from a kinematics

point of view. If an motion from the location in one state to the location in another state is not kinodynamically feasible motion, this state is never expanded. This leads to a fewer states in the state space than in the more classical MAPF approaches. SIPP is therefore considered to be suitable for realistic motion planning in which kinematics and continuous time are taken into account during planning already efficiently.

As SIPP is a low-level solver, it has to be extended to be suitable for application for multi-agent path finding. Several algorithms were discussed in literature that use SIPP for such a purpose. Unfortunately, these approaches have not been compared directly to each other. However, based on the experiments that were done in the individual papers, a trade-off between the different approaches is shown in Table 5.3 that will be further elaborated below.

Table 5.3: Comparison of motion based approaches. Minimum score is 0, maximum score is 3.

	ECBS-CT	CCBS	CCBS-kc
Performance			
Success rate	2	1	1
Runtime	3	2	1
Solution cost	2	3	3
Scalability	3	2	1
Applicability			
Large agents	2	1	1
Kinematics	2	1	3
Continuous time	3	3	3
Implementation			
Complexity	1	2	2
Availability	1	1	1
TOTAL	19	16	16

From the group of motion based approaches, experiments in [3] showed low success rate of CCBS and CCBS-kc. However, the two algorithms are the only ones of this group that return an optimal solution, which makes them score relatively high there. A comparison of the SIPPwRT is not considered as meaningful, as this algorithm was only considered for the MAPD problem and not tested for MAPE. It is therefore not included in Table 5.3. The bounded suboptimal ECBS-CT returns a solution that is a bounded factor w away from the optimal solution but it is faster in returning solutions and has a higher success rate than CCBS and CCBS-kc. In general ECBS-CT was found to be a suitable algorithm to find a solution for realistic MAMP problems [13]. However, as already described in the review of ECBS-CT, the application of the algorithm as implemented by Cohen et al. [13] is different than what is applicable for this research. The first reason is that heading angles were included in the motion feasibility and the state definition. For this research, heading is less relevant but velocity should be added to the state definition and in the consideration of the motion. An approach of evaluating motion primitives with velocity considerations was also proposed by Cohen et al. in their paper about ECBS-CT which is why it is expected that a suitable variant of ECBS-CT can be developed that takes this into account. The second reason is that the size of the reservation table used in the implementation in the paper was large due to the structuring of the environment on a cell level. It is expected that the size of the reservation table for an airport environment will be much smaller and that the algorithm will therefore be faster. With adaptations the ECBS-CT solver is therefore considered to be suitable for application in this research, under the condition that a practical working version can be developed that includes velocity considerations.

Finally, the potential of a combination of PBS and SIPP is suggested in this literature study. PBS scored high in the comparison with other more classical MAPF solvers, under the condition that the practical implications of its theoretical suboptimality should be evaluated well in experiments. The standard PBS algorithm did however not score well on applicability. At the same time, the SIPP based approaches did score high on applicability, due to their efficient way of taking continuous time into account and the option to handle kinematics efficiently. Although the combination of PBS and SIPP has not yet been considered in literature, it is expected the high level solver of PBS and SIPP as a low level (single agent) solver would be a suitable approach for planning airport ground surface operations as well. In the high-level of PBS, different priority orderings can be explored in the priority tree. The low-level then takes this priority ordering as input and performs the

path planning of the single agents respecting the priority ordering. This approach combines the efficiency of the high-level search in PBS with the efficient and continuous time friendly low-level SIPP. It should be noted that this approach should be tested in the application for planning of airport ground surface movement operations. A combination of ECBS, which received the highest score as classical MAPF solver after PBS, and a SIPP variant could be a suitable alternative for PBS + SIPP. This approach has the advantage that it has been discussed in literature before as ECBS-CT and that its solution is bounded suboptimal instead of unbounded suboptimal (as in PBS). With adaptations, ECBS-CT was therefore also considered as a suitable solver for this research. Experiments on a real airport layout will be performed to show the practical suitability of both PBS + SIPP and ECBS-CT (combination of ECBS and SIPP). Based on these experiments, a final choice between PBS+SIPP and ECBS-CT will be made.

5.6. Concluding Remarks

In this chapter, algorithms from two main groups of planning techniques were evaluated. It was shown that the "classical" MAPF approaches make assumptions that usually do not hold when applying them on real-world problems. Adaptations are thus required to make them suitable for real-world applications. Planning with continuous time, incorporating kinematics efficiently and handling actions with arbitrary durations does however remain challenging for these approaches. From the group of motion-based approaches, algorithms based on Safe Interval Path Planning (SIPP) were discussed as an efficient way to plan in continuous time. SIPP does so by defining a state as a configuration and safe interval pair rather than a location time pair. This also enables for planning with velocities efficiently. The group of SIPP-based approaches is also able to handle actions with arbitrary duration effectively. Two SIPP based approaches were concluded to be promising for application in this research. These are Priority Based Search + Safe Interval Path Planning (PBS + SIPP) and Enhanced Conflict Based Search Continuous Time (ECBS-CT). ECBS-CT will be considered if the PBS + SIPP algorithm does not prove to be suitable due to the incompleteness of PBS.

Depending on the discretization of the airport environment, the approaches need to be adapted to take into account the shapes of agents. If agents can block multiple nodes or edges at the same time, additional constraints can be introduced based on rules. This is inspired on the LA-CBS and MC-CBS approaches considered in this section. Finally, algorithms may be required to deal with the fact that new agents are constantly added to the airport system. As discussed in this chapter, the search window w used in WHCA*, CO-WHCA* and CO-WHCA*P can be used to do this.

6

Research Proposal

In this final chapter, the research proposal for this MSc thesis will be discussed. In [section 6.1](#), a recap of the literature study will first be given. In [section 6.2](#), the research objective, the research questions and the prioritisation of research questions will be discussed. The methodological steps and the scope of this research will be outlined in [section 6.3](#). Finally the research planning will be discussed in [subsection 6.3.3](#).

Before moving to the research objective and the research questions, the problem statement as identified in this literature study will be discussed, based on the concluding remarks in the end of each chapter.

6.1. Recap Literature Study

In [chapter 2](#), the conventional ground surface operations at Amsterdam Airport Schiphol were discussed. It is clear that many stakeholders are involved in this operation and that they heavily depend on Air Traffic Control to be guided over the airport safely and efficiently. With the growing number of flights, both the airport its infrastructure and the Air Traffic Control system have already reached or are reaching their maximum capacity. At the same time, developments are taking place in the area of airport ground surface operations as discussed in [chapter 3](#). TaxiBot is considered as a promising technique to reduce the emissions of taxiing operations at airports. The implementation of TaxiBot has only been tested preliminary however, in the form of single TaxiBot experiments or studies that rely on the current airport control system and procedures or use a simplified layout of the airport. In [chapter 4](#) different studies that use an agent based modelling approach to simulate a distributed ground surface operation were discussed. Although they have shown several cooperative coordination mechanisms to be effective in controlling traffic over the airport, the apron operations were excluded from these studies. Push-back processes, traffic management techniques such as gate holding and congestions around the apron area were therefore not studied before in such as system. This study aims to improve the existing models by including the apron operations and by improving the cooperative coordination mechanisms that are currently implemented. Push-back paths will be modelled in detail as well as the taxiways connected to the individual gates. Although suitable techniques were identified in [chapter 5](#), elements from these different techniques have to be combined in order to be applicable in the context of airport ground surface operations. Several motion-based approaches will therefore be evaluated for their applicability by testing them in an agent-based simulator. This model that is developed in this MSc thesis will eventually also allow for a more realistic evaluation of the TaxiBot concept in a distributed multi-agent setting and a comparison to current operations.

6.2. Research Objective and Research Questions

Based on this literature study, the following research objective is formulated:

”To design and evaluate a distributed multi-agent control system for a novel concept of autonomous electric taxiing operations by focusing on apron operations and the implications thereof.”

From the research objective, the following research questions and sub questions were identified. After introducing the research questions, the prioritisation of the questions will be discussed. The experiments that will be performed to answer these questions will then be described in [subsection 6.3.1](#).

1. How does a distributed multi-agent control system, including aprons operations, affect the performance of airport ground surface operations with respect to a system that excludes apron operations?

- (a) What is the effect of push-back and engine start on blocking times of taxiways and aprons?
- (b) How does the limited capacity of gates, aprons and holding positions affect the performance of the ground surface operations?

The distributed multi-agent control system will be based on a variant of the Safe Interval Path Planning (SIPP) algorithm in combination with either PBS or ECBS-CT. The performance of the airport ground surface operation will be evaluated based on the following key performance indicators (KPI's):

- Taxi time.
- Taxi distance.
- Congestion (based on heatmaps).
- Waiting times (inbound, outbound and total).
- Introduced holding delay for outbound flights.
- Number of gate conflicts (only in layout with gates).

2. How does the performance of a distributed multi-agent control system, that includes apron operations compare to the performance of conventional operations?

The distributed multi-agent control system is the same as in question 1. Also KPI's used to evaluate performance in this second question are the same as in question 1.

3. To what extent can outbound holding, enabled by the distributed multi-agent control system, improve the performance of the airport ground surface movement operation?

The distributed multi-agent control system is the same as in question 1. An additional KPI that measures the number of outbound hold actions is added to the previously mentioned KPI's to evaluate the performance of the airport ground surface movement operation with outbound holding.

4. How does the implementation of autonomous engine-off taxiing with TaxiBots affect the performance of airport ground surface movement operations on an apron level, in comparison to the conventional operations (both in distributed control system)?

The distributed multi-agent control system is the same as in question 1. An additional KPI that measures the time spend on the apron will be added for this fourth research question. This KPI is calculated as follows:

- For outbound flights: Time spend on apron = apron exit time - stand exit time.
- For inbound flights: Time spend on apron = stand entry time - apron entry time.

5. To what extent can outbound holding, enabled by a distributed multi-agent control system with autonomous TaxiBots, improve the performance of the airport ground surface movement operations?

6.2.1. Prioritisation of research questions

The above mentioned research questions will be prioritised as follows. Firstly, question 1 will be answered. By answering this question, the effects of the addition of apron operations to the agent based simulator will be clear. It is a prerequisite for answering questions 3, 4 and 5. Secondly, question 2 will be answered. The answer to this question shows how the concept of distributed multi-agent control on a simulated airport layout with apron and gates relates to the actual ground surface movement operation. It will therefore add additional insights (with respect to previous studies that considered distributed multi-agent control) in the suitability of distributed multi-agent control for airport ground surface movement operations. Question 3 will be answered third. It should be noted that this is under the condition that double holding delays occur. The results of the research experiments performed for research question 1 have to show how often the double holding delay actually occurs. If shown to increase waiting times significantly, outbound holding will be tested. If not, this research question will be moved lower in priority and question 4 will be answered third instead. Question 4 will be answered fourth. After question 1 and 2 have been answered, TaxiBots can be added to the model. Finally, question 5 will be answered. It makes sense to test the concept of outbound holding with TaxiBots only when question 3 and question 4 have been answered.

6.3. Research Methodology

In this section, the research methodology will be discussed briefly. First, the methodological steps that will be followed to answer the research questions are discussed on a high level. Then the research scope will be defined to further specify the concept of operations used in this study. Finally, the high and low level tasks will be placed in time to form a research planning.

6.3.1. Methodological Steps

The high level steps that will be followed in order to answer the research questions will be described in this section. The planning is then shown in [Table 6.1](#).

Conceptualisation

Before a model is developed, the current operations will be evaluated in more detail. Restrictions on the use of taxiways, gates and aprons at Amsterdam Airport Schiphol will be converted. Also push-back paths will be defined in more detail based on the standardized paths as shown in [Figure 2.11](#) in [subsection 2.2.3](#).

Besides that, a conceptual agent-based model will be developed in the conceptualisation. This will first be done for the baseline model and then also for the extended model. The environment, agents and interactions between agents will be specified in this conceptual model. This model will then be formalized to be implemented as the first baseline model.

Baseline Model

First, a baseline model will be developed based on the previous research projects that were done at ATO in the direction of agent-based airport ground surface modelling. Together with one other MSc student and one PhD student, a novel, modular, agent based simulator will be developed. In the first version of this simulator, gates will be simplified and only a limited number of aircraft will be modelled. PBS + SIPP will be implemented and tested in this model. It will be evaluated to what extent this motion-based planning technique is able to take into account kinematics during planning efficiently. If not proven to be suitable, ECBS-CT + SIPP will be implemented as alternative as concluded in [chapter 5](#).

Extended model

The baseline model will be extended in three steps. Firstly, gates and aprons will be introduced which allows for the inclusion of apron operations. Secondly, outbound holding will be introduced and finally TaxiBot operations will be added. Each extension will be validated and verified based on expert opinion and by using test scenarios. Below, the relation of the extensions to the research questions is discussed.

Extension 1: Addition of gates and aprons

At this point, simulation experiments related to research question 1 and 2 can be performed. This will be done in the following way to answer research question 1. The output of two simulation scenarios will be compared. The first scenario will be based on an airport layout without gates. Meta-gates will be used as a simplification of the apron and gates in this scenario. The second scenario will be based on an airport layout with gates and

aprons. In this second scenario, apron operations will thus be included. Other simulation parameters will remain unchanged between the two experiments. For research question 2, simulation outputs will be compared to actual operational data. Operational data is based on A-CDM milestone data for the Actual Landing Time (ALDT), Actual Take Off Time (ATOT), Actual Off Block Time (AOBT) and Actual In Block Time (AIBT). These milestones can be used to calculate the taxi in time and taxi out time.

Extension 2: Addition of outbound holding

The extension of the model for outbound holding does not require changes to the layout or the addition of new vehicles. At this point, research question 3 can be answered. Two scenarios will again be compared. In one scenario, an additional degree of freedom is added to the planning. This enables the planner to switch around an inbound aircraft that is waiting for a gate and an outbound aircraft that is waiting (but ready) to depart. In the other scenario, this additional freedom is not part of the planning options. Other simulation parameters will remain constant.

Extension 3: Addition of TaxiBot

In the third extended model, TaxiBots will be added to the model. Simulation experiments related to research question 4 and 5 are performed at this point. Research question 4 will be answered by comparing simulation outputs of a scenario with the conventional push-back and taxi operation to the simulation outputs of a scenario with TaxiBot operations. Other simulation parameters will remain constant. To answer research question 5, similar experiments as described in extension 2 will be performed. TaxiBots will be included in this last case.

Final evaluation and analysis

The resulting model will be analysed based on additional simulation scenarios. Sensitivity analysis will then be performed as well as a statistical analysis.

6.3.2. Research scope

In order to narrow down this research, this section will cover the assumptions that are made in this study. These are subdivided in several categories. Note that these assumptions will be further extended during the development of the conceptual model.

General Assumptions

- The layout of Amsterdam Airport Schiphol will be used as a case study.
- Non-nominal operations such as de-icing operations or low visibility conditions are not considered.
- Deviation from the standard taxi directions at Amsterdam Airport Schiphol is allowed.
- Future expansion of taxiway Quebec is completed.
- Departure and arrival sequence is given.
- The schedule used as input for this study contains an arrival time at the airport and a departure time from the airport.
- The schedule used as input for this study contains an origin and destination in the form of a (runway entry, gate) pair or (gate, runway entry) pair.
- Aircraft categories are simplified into size categories
- Humans are excluded from the system, commands are assumed to be adhered to by all agents moving over the airport.

Apron Assumptions

- Wingspan restrictions apply as defined in the AIP.
- Push-back paths are modelled based on the standard push-back paths.
- Time needed for push-back is calculated from the standard push-back path and a push-back speed.
- Aircraft can only push back when their entire push-back path is free.
- A fixed time will be assumed for engine start at the apron.

- Gate assignment is assumed given.
- Gates will be blocked for a fixed time per aircraft to represent the turn around at the gate.
- No delays or disruptions caused by ground handling at the gate are considered.

TaxiBot Assumptions

- TaxiBots will only be involved in the simulation experiments related to research question 4 and 5.
- Only focus on TaxiBot, not on wheeltug or single engine on taxiing.
- Only outbound aircraft use TaxiBots, inbound aircraft taxi on their own power
- Task assignment will be excluded, TaxiBot is available for each aircraft.
- TaxiBots are assumed to be autonomous.
- Unlimited availability of TaxiBots is assumed.
- Engine warm-up can be done during Taxiing
- Decoupling locations of TaxiBot are assumed to be given and are fixed per runway.
- As long as two aircraft+TaxiBots are not in conflict, simultaneous operations are allowed.
- Battery life of TaxiBot is assumed to be sufficient to operate a day without charging.
- Fixed decoupling times will be assumed.
- TaxiBots can use service roads around the apron area to move to the gate.

6.3.3. Planning

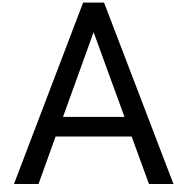
The research planning is shown in [Table 6.1](#) with a subdivision in high level tasks and low level tasks.

Table 6.1: Research Planning

High level task	Low level tasks	Duration
Start Up	Familiarize with existing tools	2 weeks
Conceptualisation	Define conceptual model	4.5 weeks
Implementation baseline model	Develop simulator Develop baseline airport layout Implement motion based baseline planner	4 weeks
Test baseline model	Develop test scenarios Test model with test data Validate and verify baseline model	3.5 weeks
Implement first extended model	Add gates, aprons and holding areas to layout Update motion based planner to work with gates and aprons	2 weeks
Test first extension model	Develop test scenarios Test model with test data Validate and verify extended model	1.5 weeks
Implement second extended model	Add TaxiBots Update planner to work with TaxiBots	2 weeks
Test second extension model	Develop test scenarios Test model with test data Validate and verify extended model	1.5 weeks
Add extensions	Analyse model Potentially introduce new techniques, improve model	2.5 weeks
Final evaluation and analysis	Develop simulation scenarios Perform simulations Perform sensitivity and statistical analysis Write draft	7.5 weeks
Conclude thesis work	Write final paper Prepare presentation and defence	5 weeks

III

Supporting work



Model Elaboration

In this chapter, the model will be further elaborated. In [section A.1](#) we explain the construction of the graph layout and the relevant traffic rules at AAS. Then, in [section A.2](#) we provide more details on the motion generation and pseudocode for the relevant algorithms. Finally, in [section A.3](#), we show a selection of the verification steps performed.

A.1. Layout Graph Generation Amsterdam Airport Schiphol

This section elaborates on the layout of Amsterdam Airport Schiphol (AAS). In [subsection A.1.1](#) we first supply background information about the runways naming and locations of the runways at AAS. In [subsection A.1.2](#) we then explain the creation of the abstracted graph layout used in the MAS, including the creation of push-back paths and decoupling locations.

A.1.1. Runway Placement at AAS

Before going into the creation of the layout graph used in this study, we give some background information for readers unfamiliar with the layout of Amsterdam Airport Schiphol (AAS) in this subsection.

[Figure A.1](#) shows the six runways at AAS. In this study, we study schedules with two runway modes of operations (RMO's). During RMO North, shown in [Figure A.1](#), the Polderbaan and Zwanenburgbaan are used for take-offs, whereas the Kaagbaan and Aalsmeerbaan are used for landings. During RMO South, shown in [Figure A.2](#), flights depart from the Kaagbaan and the Aalsmeerbaan, and flights land at the Polderbaan and the Zwanenburgbaan.

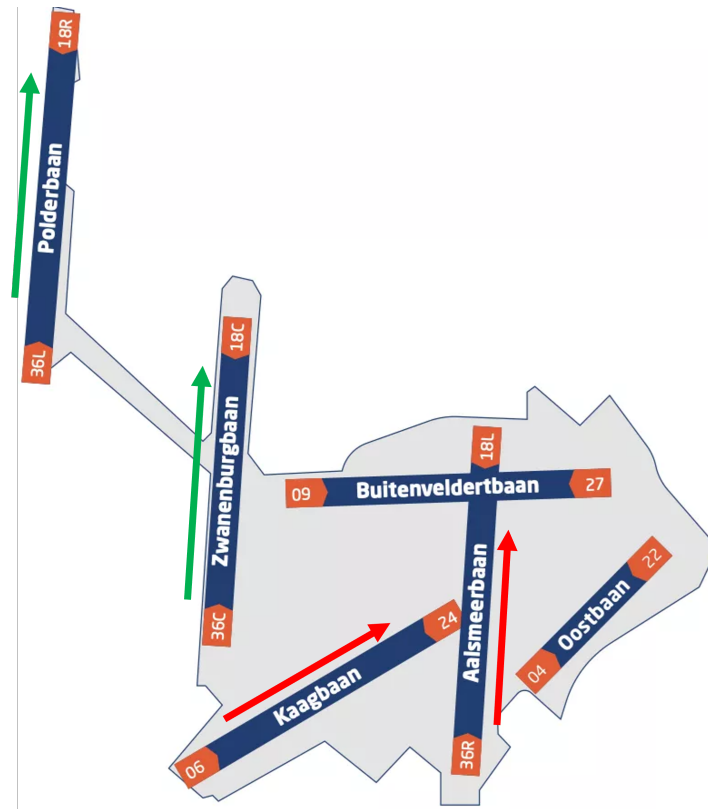


Figure A.1: RMO North Runways. Green: departure runways, Red: arrival runways. Adapted based on LVNL [48]

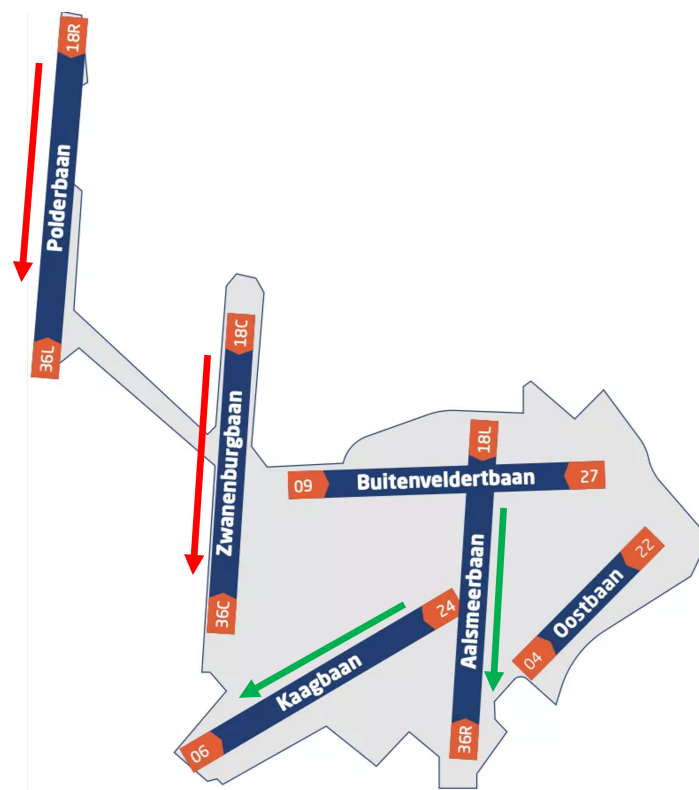


Figure A.2: RMO South Runways. Green: departure runways, Red: arrival runways. Adapted based on LVNL [48]

A.1.2. Construction of Abstracted Graph Layout

To represent the airport layout in the model, an abstracted graph was created. The locations of the runways, taxiways, and ramps were derived from X-Plane developer airport data files. Taxiway center lines and runways were translated into edges. A node was created at every point where two or more edges meet or to split very long edges into segments. Furthermore, ramps were translated into nodes. In the creation of this initial version of the airport layout, the locations of edges and nodes was extensively verified based on satellite data and operational maps. The airport data file was then extended in multiple steps. First, a ramp edge was connected to the ramp node to represent the parking position of the aircraft. Second, entry and exit edges were added in order to connect the ramp edges to the taxiway edges. Third, edges and nodes were labelled with an ID so that ramp names, decoupling locations, restricted edges, and runway entries could be mapped to the correct node in the graph. The result is shown in [Figure A.3](#).

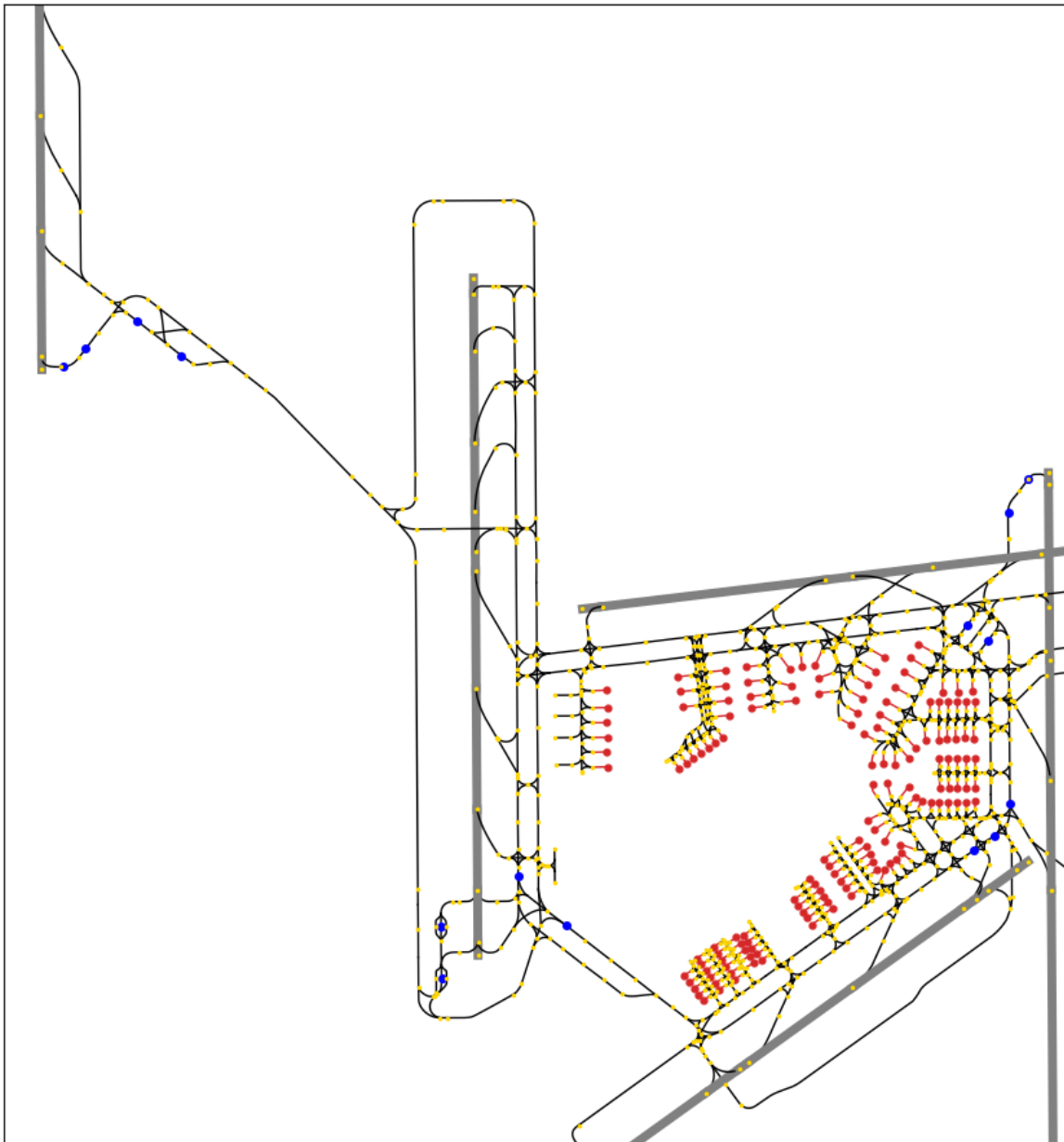


Figure A.3: Part of the layout with: regular nodes (yellow), ramp nodes (red), ramp edges (red), taxiways (black), decoupling locations (blue), runways (grey).

Push-back and Push-pull paths

This section further explains the creation of the push-back and push-pull paths in the model. It was assumed that aircraft always use the standard push-back path as described by Amsterdam Airport Schiphol [75]. For some gates, the path is dependent on the aircraft type. If this was the case, we created multiple paths and labelled each path with the compatible ICAO types on that path (e.g. one path for ICAO-A, ICAO-B, and ICAO-C, and another path for ICAO-D, ICAO-E, and ICAO-F). The created push-back path starts at the ramp node and then consists of a series of directional edges to be followed by the aircraft in backwards direction. The pull-path is followed in forward direction. Figure A.4 shows an example of the modelled push-back and push-pull path based on the original paths shown in Figure A.5.

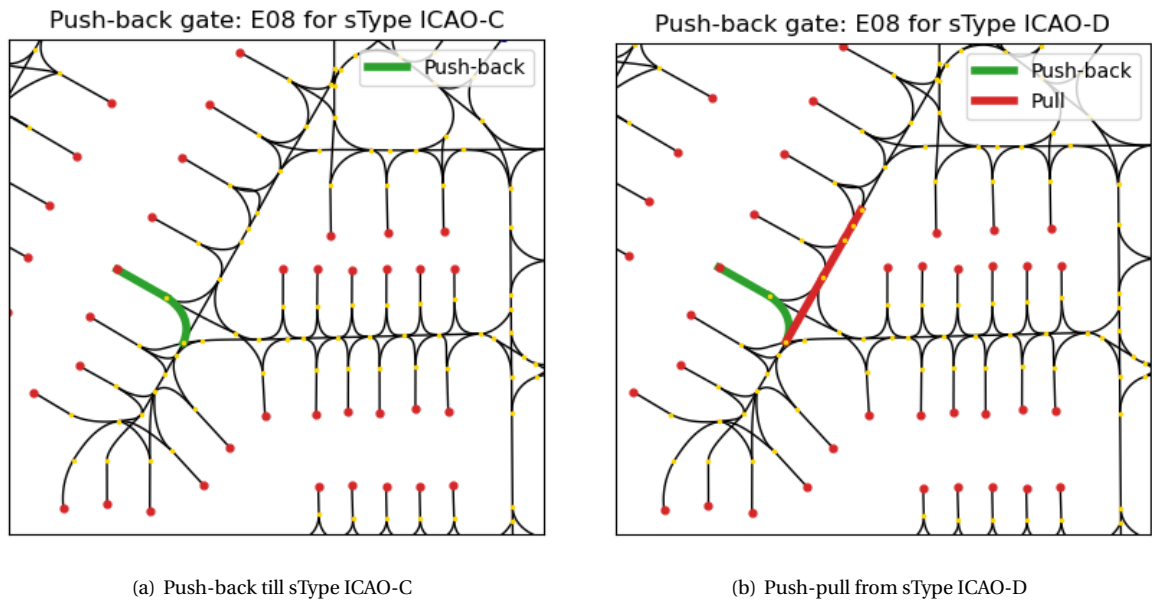


Figure A.4: Example push-back path from ramp E08 for two different aircraft types.

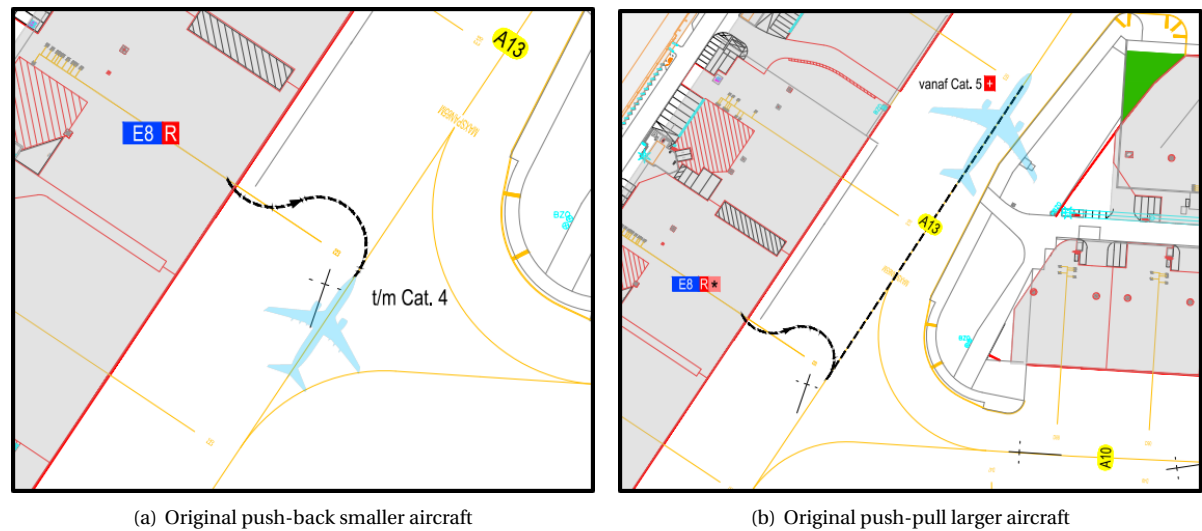
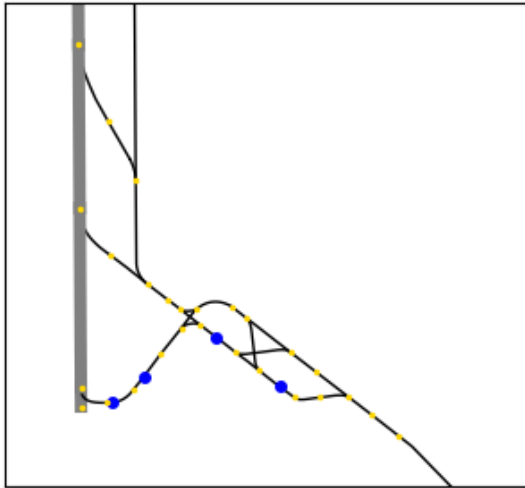


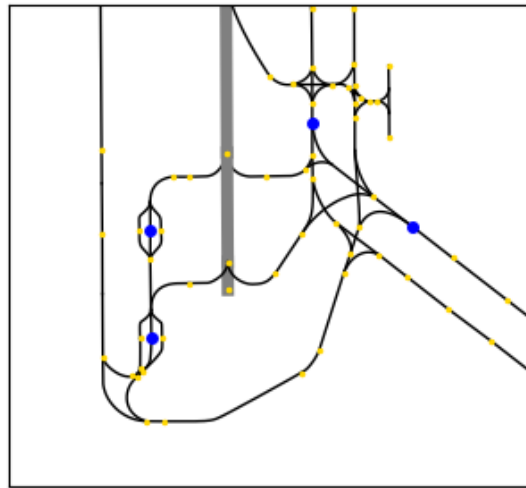
Figure A.5: Original push-back path from ramp E08 for two different aircraft types. Obtained from [75]

Decoupling Locations

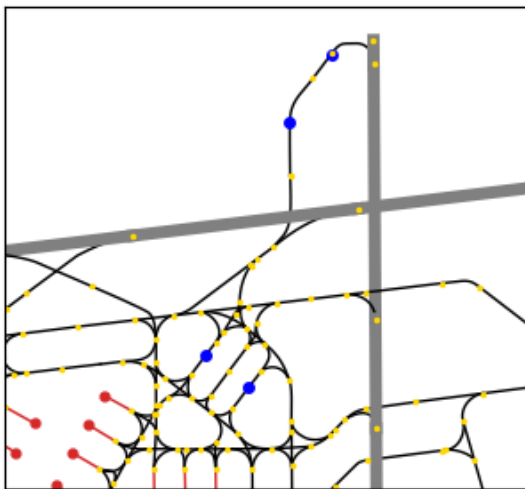
Decoupling locations for this research were based on a subset of the decoupling locations in another MSc Thesis by J.A.L. Soomers [91] who compared several decoupling configurations at AAS. Soomers defined 15 decoupling points together with AAS with their criteria for uncoupling locations. In this thesis, we use all points that were defined per runway as possible decoupling locations, as shown in Figure A.6. It should be noted that not all criteria are fully met with the selected points. However, it is expected that future developments of the tug, protocols, and the airport infrastructure will make the locations feasible.



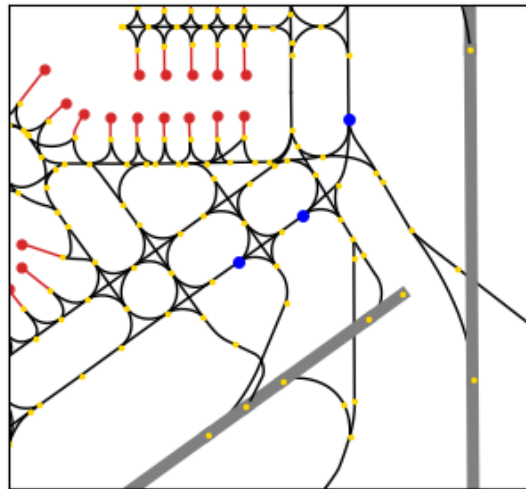
(a) RWY 36L



(b) RWY 36C



(c) RWY 18L



(d) RWY 24

Figure A.6: Decoupling locations per runway (RWY) shown as blue node.

Relevant Traffic Rules during Runway Configurations

For safety reasons, some taxiways cannot be used when certain runways are active at AAS. Those are the taxiways that cross an active runway or that are located on a part that is flown over by a landing or departing aircraft. Figure A.7 shows the relevant edges that can be blocked under the Runway Mode of Operations (RMOs) considered in this thesis. The edges with a red cross are blocked when runway 18C/36C (Zwanenburg Runway) is used for take-offs in the Northern direction (towards top of the image) or for landings in the Southern direction (towards bottom of the image). This is the case for RMO North and RMO South respectively. All aircraft taxiing towards or from runway 36L/18R (Polderbaan) are then forced to pass the Zwanenburg Runway on the South side. When the Zwanenburg runway is not active, the edges with the red cross are not constrained and can be used by all traffic.

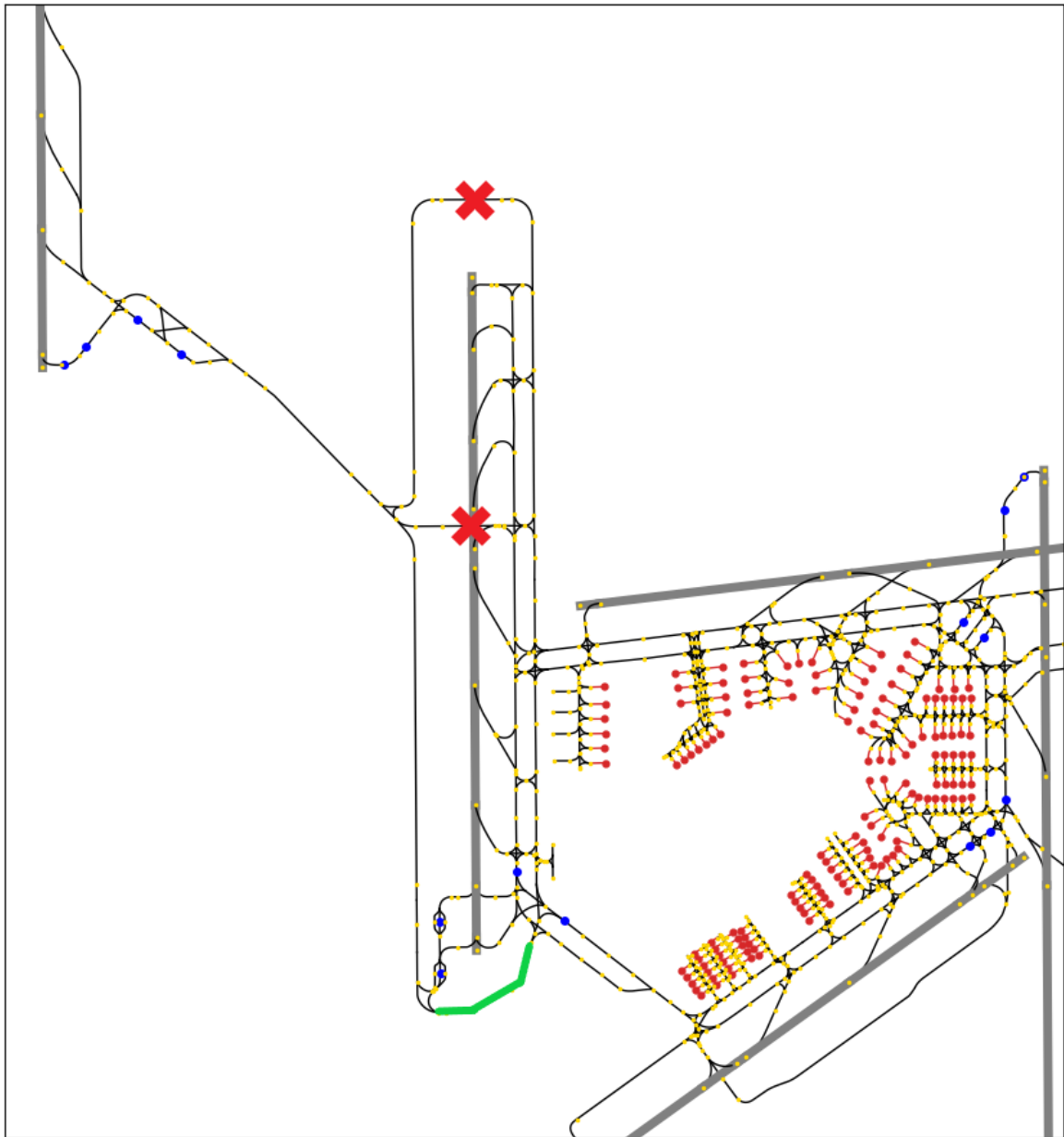


Figure A.7: Two blocked taxiways (red crosses) when Runway 36C is active for departures under RMO North or when Runway 18C is active for landings under RMO South. All aircraft to or from 36L/18R are directed over the taxiways south of Runway 36C/18C as shown in green. Northern direction is towards top of the image.

A.2. Route Planning

In this section, we elaborate on the route planning as performed by the Routing Agent. In [subsection A.2.1](#), the activity sequences for both Multi-Engine Taxiing and Tug-Enabled Taxiing operations are explained. Next, in [subsection A.2.2](#) we discuss the PBS algorithm and in [subsection A.2.3](#) the SIPP algorithm is explained. Finally, we elaborate on the motion generation in [subsection A.2.4](#).

A.2.1. Activity Sequences

In the main paper it was already shown that the Routing Agent defines an activity sequence with the *Create Activity Sequence Property*. It uses three building blocks: the 1) Go-to activity, 2) Follow activity, and 3) Wait activity. In this section, we show how these activities are combined in Multi-Engine Taxiing and Tug-Enabled Taxiing below.

Activity Sequences for Multi-Engine Taxiing

For departing aircraft using Multi-Engine Taxiing, the activity sequence is dependent on how aircraft leave the gate. In case of a push-back only, we define the following three activities:

1. **Follow:** drive backwards to the end of the push-back path.
2. **Wait:** wait for decoupling time of the push-back truck, wait for remaining engine start up time, and change direction.
3. **Go-to:** go to one of the goal nodes on the runways.

In case the aircraft departs from a gate that requires a push-pull manoeuvre, we define two additional activities as shown below. Note that the entire engine start is now performed after finishing the pull. The sequence is defined as follows for this case:

1. **Follow:** drive backwards to the end of the push-back path.
2. **Wait:** wait for t_{switch} and change direction.
3. **Follow:** drive forwards to the end of the pull path.
4. **Wait:** wait for entire time needed to start the engines.
5. **Go-to:** go to one of the goal nodes on the runways.

In case the aircraft departs from a taxi-out gate, it does not require the assistance of a push-back truck. The activity sequence is defined as follows in this case:

1. **Go-to:** go to one of the goal nodes on the runways. Note that aircraft hold at their ramp till the engines are started, otherwise they cannot move.

Activity Sequences for Tug-Enabled Taxiing

For outbound aircraft that use Tug-Enabled Taxiing, operations are different as was already explained in the main paper. Started engines are not required for taxiing and the pull path is skipped. The activity sequence is defined as follows:

1. **Follow:** drive backwards to the end of the push-back path.
2. **Wait:** wait for t_{switch} and change direction.
3. **Go-to:** go to one of the decoupling locations for the departure runway.
4. **Wait:** decouple and wait for remaining engine start up time.
5. **Go-to:** go to one of the goal nodes on the runways.

Note that, for using Tug-Enabled Taxiing from taxi-out gates, activity 1 and 2 are not required as the aircraft-tug combination can leave the gate nose first.

A.2.2. Priority Based Search

The algorithmic representation of Priority Based Search (PBS) [52], the high-level algorithm used in this study is provided below in Algorithm 1. Changes with respect to the original PBS algorithm by Ma et al. are indicated in red. The supporting functions *UpdatePlan* and *getReservations* are presented in Algorithm 2 and Algorithm 3, respectively. A graphical representation of the USI merging in *getReservations* is shown in Figure A.8.

Algorithm 1 Priority Based Search with Root Constraints. Adapted version of PBS [52], changes/additions indicated in red.

```

1: Input MAPF instance, rootConstraints
2: STACK =  $\emptyset$ 
3: N.plan =  $\emptyset$ 
4: N.rootConstraints = rootConstraints                                     // Initialize root constraints
5:
6: // Calculate Root Solution and Avoid Root Constraints
7: for each agent  $a_i$  do
8:   success = UpdatePlan(N,  $a_i$ )
9: end for
10:
11: if not success then
12:   return No Solution
13: end if
14: N.Cost = sum of taxi times of all paths
15: STACK += N                                                         // Add root node to STACK
16:
17: // High-Level Search
18: while STACK  $\neq \emptyset$  do
19:   N = pick and remove top node from STACK
20:   if N.plan is collision free then
21:     return MAPF solution
22:   end if
23:   C = first collision in time in N.plan
24:   N.children =  $\emptyset$ 
25:
26:   for each  $a_i$  in C do
27:     N'.plan = N.plan
28:     N'.rootConstraints = N.rootConstraints                             // Pass route constraints to child node
29:     N'.priorities +=  $a_i > a_j$                                          // Add priority pair
30:     success = UpdatePlan(N',  $a_j$ )
31:     if success then
32:       N.children += N'                                               // Only add child when solution was found
33:     end if
34:   end for
35:   Insert N.children into STACK in order of decreasing cost (child with lowest cost on top of STACK).
36: end while
37:
38: return No Solution

```

Algorithm 2 UpdatePlan(N, a_i) Adapted version of PBS [52], changes/additions indicated in red.

```

1: Input PBS search node  $N$  with priorities, rootConstraints, and plans, agent  $a_i$ 
2: LIST = topologicalSort( $N.priorities.Down(a_i)$ ) // Get all agents lower in priority

3: for each agent  $a_j$  in LIST do
4:   if  $a_j == a_i$  OR  $a_j$  is in conflict with any higher prioritized agent then
5:      $N.graphReservations(a_j) = \text{getReservations}(a_j, N.priorities.Up(a_j), N.solution, N.rootConstraints)$ 
6:      $\text{success}, N.plan = \text{doSIPP}(N.graphReservations(a_j), a_j)$ 
7:   end if
8:   if not success then
9:     return False // No single agent path is found
10:  end if
11: end for
12: return True // Single agent path is found

```

Algorithm 3 getReservations($a_i, N.priorities.Up(a_i), N.solution, N.rootConstraints$)

```

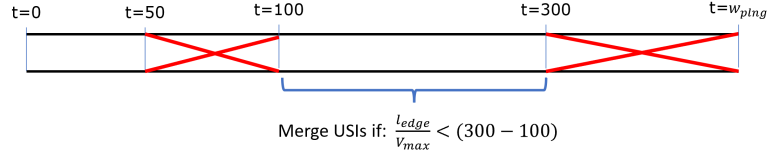
1: Input Agent  $a_i$ , higher prioritized agents, solutions of higher prioritized agents, and root constraints
2: USIs =  $\emptyset$ 
3: Restrictions =  $\emptyset$ 
4: No-go edges =  $\emptyset$ 
5:
6: // Collect all USIs and Restrictions from Routing Agent and from higher prioritized agents
7: for each edge  $E$  do
8:   USIs[ $E$ ] =  $\emptyset$ 
9:   Restrictions[ $E$ ] =  $\emptyset$ 
10:
11:   USIs[ $E$ ] += USIs set by RoutingAgent on edge  $E$  for  $sType(a_i)$ 
12:   Restrictions[ $E$ ] += Restrictions set by RoutingAgent on edge  $E$  for  $sType(a_i)$ 
13:
14:   for each agent  $a_j > a_i$  do
15:     USIs[ $E$ ] += USIs set by agent  $a_j$  on edge  $E$  for  $sType(a_i)$ 
16:     Restrictions[ $E$ ] += Restrictions set by agent  $a_j$  on edge  $E$  for  $sType(a_i)$ 
17:   end for
18:
19:   for each USI in USIs[ $E$ ] do
20:     if time between USI and next USI < length( $E$ ) /  $V_{max}$  then
21:       merge USI and next USI
22:     end if
23:     if USI[ $E$ ] ==  $[t_{start}, t_{start} + w_{plng}]$  then
24:       No-go edges +=  $E$ 
25:     end if
26:   end for
27: end for
28:
29: // Create SIs
30: for each edge  $E$  do
31:   SIs[ $E$ ] = set of non-overlapping intervals within  $[t_{start}, t_{start} + w_{plng}]$  that do not overlap with USIs
32: end for
33:
34: return graphReservations // contains USIs, restrictions and No-go edges

```

1. Check USIs set by agents higher in priority than a_i and USIs set by the Routing Agent

USIs on edge E from each higher prioritized agent $a_j > a_i$		USIs on edge E from the Routing Agent	
sType	USIs duration	sType	USIs duration
ICAO A	[(50,100), DNE]	ICAO A	[(300, inf), DNP]
...	[...]	...	[...]
ICAO F	[(40, 110), DNE]	ICAO F	[(300, inf), DNP]

2. Collect USIs for only for the sType of agent a_i (assume ICAO A here)



3. Construct Safe Intervals on edge E for agent a_i and store DNP USIs

$$SIs[E] = [(0,50), (100,300)]$$

$$DNP[E] = [(300, w_{plng})]$$

Figure A.8: Example USI merging. Middle image under 2. based on image from Phillips and Likhachev [58].

A.2.3. Safe Interval Path Planning

Algorithm 4 Safe Interval Path Planning. Adapted version of SIPP [58], changes/additions indicated in red.

- 1: **Input** set of possible final goal configurations s_{goal} , activity sequence, graphReservations, agent kinematics
- 2: $OPEN = \emptyset$
- 3: $g(s_{start}) = 0$
- 4: $f(s_{start}) = g(s_{start}) + h(s_{start})$
- 5: $OPEN += s_{start}$
- 6: **while** s_{goal} not expanded **do**
- 7: s = state with minimum f -score from $OPEN$
- 8: **successors** = **getSuccessors**(s)
- 9: **for** s' in **successors** **do**
- 10: **if** s' not visited before **then**
- 11: $f(s') = g(s') = \infty$
- 12: **end if**
- 13: **if** $g(s') > g(s) + c(s, s')$ **then**
- 14: $g(s') = g(s) + c(s, s')$
- 15: update arrival time in s'
- 16: **end if**
- 17: $f(s') = g(s') + h(s')$
- 18: Insert s' in $OPEN$ with $f(s')$
- 19: **end for**
- 20: **end while**
- 21: **return** path

Algorithm 5 getSuccessors(s)

```

1: Input state  $s$ 
2: Successors =  $\emptyset$ 
3:  $E = (N, N_n)$  // edge in state  $s$ 
4: ttExceedOvertake = getOvertake(E) // check if a time to exceed to prevent overtaking is defined

5: for next edge in neighbors(E) do
6:   if next edge in no go edges then
7:     continue // do not generate successor if next edge is no go edge
8:   end if

9:   // Regular Successor Generation
10:  for SI on next edge do
11:    ttExceedSI = getSI(tt(SI)) // check if time to exceed to make start of SI is defined
12:    ttExceed = max(ttExceedSI, ttExceedOvertake)
13:     $V_{lim} = \text{getVlim}(\text{next edge})$  // get limit velocity on next edge

14:     $t_{trav}, v_f = \text{getMotions}(\text{ttExceed}, V_{lim}, \text{length}(\text{next edge}))$  // determine traversal time and final velocity
15:     $t_{arr} = t_{start} + t_{trav}$ 

16:    if  $t_{arr} > \text{endSI}$  then
17:      continue // continue if fastest motion results in later arrival time than end of the SI
18:    end if

19:    if obligation to proceed on E and  $[t_{start}, t_{arr}]$  overlaps with obligation then
20:      continue // continue if there is an obligation to proceed that overlaps with motion
21:    end if

22:    if DNP USI on E and  $[t_{start}, t_{arr}]$  overlaps with DNP USI then
23:      continue // continue if there is a Do Not Persist constraint that overlaps with motion
24:    end if

25:     $s' = \text{state with } \text{cfg}(N, N_n, v_f, \text{act}_{ID}) \text{ and SI next edge}$ 
26:    Successors +=  $s'$ 
27:    RegularSuccessors +=  $s'$ 
28:  end for

29:  // Anticipatory Successor Generation
30:  for  $s^*$  in RegularSuccessors do
31:    BrakingDistance = calcBrakeDist( $s^*$ ) // calculate braking distance from state  $s^*$  to full stop
32:    UnfeasibleSet =  $\emptyset$ 

33:    for each edge within the BrakingDistance do
34:      UnfeasibleSet += getUnfeasibleSuccessors( $s^*$ ) // determine motions that cannot be made from  $s^*$ 
35:    end for

36:    for UnfeasibleSuccessor in UnfeasibleSet do
37:      success = try to generate motion from expanded state  $s$  to reach UnfeasibleSuccessor
38:      if not success then
39:        continue // if UnfeasibleSuccessor can not be made from input state  $s$ , skip this successor
40:      end if

41:       $s^{*'} = \text{successor state created from } s$ 
42:      Successors +=  $s^{*'}$ 
43:    end for
44:  end for
45: end for

46: return Successors

```

A.2.4. Motion Generation

In the paper it was already explained that we assume that vehicles move to a next edge as quickly as possible, unless there is an USI or restriction that prevents the vehicle to do so. In this section, we further explain how motions are generated during path planning. To generate a motion, we use the following inputs:

- v_i : **the initial velocity** in the start state that SIPP expands.
- v_{max} : **the maximum possible velocity** for a vehicle in the current activity.
- a : **the acceleration or deceleration rate** in the current activity
- v_{limit} : **the maximum final velocity** at the end of the motion.
- t_{exceed} : **the traversal time to exceed** on an edge.
- $t_{maxhold}$: **the maximum holding time** is the maximum time a vehicle can wait in the expanded state.

Depending on the input values, we then distinguish between three cases: A) No limit velocity is defined, B) a limit velocity is defined, and C) a traversal time to exceed is defined. The motion generation in all three cases is further discussed below, after first introducing two blocks with theoretical background below:

General Formulas for Constant Acceleration / Deceleration

When a vehicle accelerates or decelerates on the entire edge, v_f is calculated by using [Equation A.1](#). The corresponding distance traversed during the accelerating or decelerating motion is computed with [Equation A.2](#).

$$v_f = v_i + a \cdot t_{acc} \quad \text{in [m/s]} \quad (\text{A.1})$$

$$d = v_i \cdot t_{acc} + \frac{1}{2} \cdot a \cdot t_{acc}^2 \quad \text{in [m]} \quad (\text{A.2})$$

However, t_{acc} in the above equations is often unknown, whereas the edge length d , initial velocity v_i , and acceleration rate a are known. Therefore, we rewrite the above equations. Solving for t_{acc} in [Equation A.2](#) gives [Equation A.3](#), which can be substituted into [Equation A.1](#) to get [Equation A.4](#) for final velocity when the initial velocity v_i , distance d , and acceleration rate a are known.

$$t_{acc} = \frac{-v_i}{a} + \frac{\sqrt{v_i^2 + 2ad}}{a} \quad \text{in [s]} \quad (\text{A.3})$$

$$v_f = \sqrt{v_i^2 + 2ad} \quad \text{in [m/s]} \quad (\text{A.4})$$

If the final velocity v_f is already known, we can use [Equation A.5](#) to calculate the t_{acc} .

$$t_{acc} = \frac{v_f - v_i}{a} \quad \text{in [s]} \quad (\text{A.5})$$

General Formulas for Partial Acceleration / Deceleration and Partial Constant Velocity

For vehicles that partly accelerate or decelerate on an edge and partly hold a constant velocity, the acceleration or deceleration distance is calculated with [Equation A.2](#), where t_{acc} is the time that the vehicle accelerates or decelerates at constant acceleration rate a . The edge length l is expressed as in [Equation A.6](#), with t_{const} is the time the vehicle holds constant velocity v_{const} . Rewriting this equation gives [Equation A.7](#), which can be used to calculate the time at constant velocity.

$$l = v_{const} \cdot t_{const} + d_{acc} \quad \text{in [m]} \quad \text{where: } v_{const} = \max(v_i, v_f) \quad (\text{A.6})$$

$$t_{const} = \frac{l - d_{acc}}{v_{const}} \quad \text{in [s]} \quad \text{where: } v_{const} = \max(v_i, v_f) \quad (\text{A.7})$$

The traversal time on the edge is then calculated with [Equation A.8](#).

$$t_{trav} = t_{acc} + t_{const} \quad \text{in [s]} \quad (\text{A.8})$$

A) $v_i < v_{max}$ and no limit velocity

When the initial velocity is lower than the maximum velocity and no limit velocity is defined, the algorithm first calculates the maximum final velocity v_f reachable from the start state of the motion. This is done by using Equation A.4 with distance d equal to the edge length. When the resulting $v_f > v_{max}$, the algorithm sets $v_{limit} = v_{max}$ and proceeds to step B). Else, if $v_f \leq v_{max}$, the vehicle can constantly accelerate on the edge. The traversal time is then calculated by using Equation A.3.

B) v_{limit} defined and $v_{limit} < v_{max}$

A v_{limit} can be defined on an edge for multiple reasons: a turn velocity, the end of an activity, or as set in step A) of the motion generation. We distinguish between three possible scenarios in this case: $v_i = v_{limit}$, $v_i < v_{limit}$, and $v_{limit} < v_i$. Each will be elaborated below.

- If $v_i = v_{limit}$, the only possible motion is to hold v_i , as we do only allow for monotonic profiles on an edge.
- If $v_i < v_{limit}$, thus a limit velocity defined that is larger than the initial velocity, the algorithm first checks whether $v_{limit} < v_f$, as calculated in A). If this is true, the vehicle can constantly accelerate on the entire edge. The traversal time is then calculated by using Equation A.5 with $v_f = v_{limit}$. Else, if $v_{limit} < v_f$, a velocity profile that consists of an acceleration part and a constant part should be generated. In this case, the algorithm calculates t_{acc} using Equation A.5 with $v_f = v_{limit}$. Given the time the vehicle accelerates, the acceleration distance d_{acc} is computed with Equation A.2. Subsequently, Equation A.7 is used to calculate the time at constant velocity. This time is then used to calculate the traversal time with Equation A.8.
- If $v_{limit} < v_i$, the similar calculations as in the previous step are performed with a equal to the deceleration rate. In this case, we first check if the v_{limit} can be achieved by decelerating over the entire edge length, by using Equation A.4. If not, there is no possible motion given the specified edge length, initial velocity, and limit velocity.

As the algorithm optimizes for shortest traversal time, v_{const} is always chosen to be the maximum of the initial velocity v_i and the limit velocity v_{limit} . This means, acceleration will take place at the start of the edge, whereas deceleration will take place at the end of an edge as much as possible. However, when a minimum traversal time to exceed is defined this is not always desirable as will be discussed next.

C) Minimum traversal time t_{exceed} is defined

A minimum traversal time to exceed can be defined on an edge when the vehicle is not allowed to enter the next edge before a certain time point, for example due to an USI on the next edge. If this is the case, we perform the same calculations as in A) and B), depending on whether a v_{limit} is defined or not. Given the traversal time t_{trav} from these calculations, we check whether t_{trav} is larger than or equal to t_{exceed} . If so, the motion is valid. If not, the algorithm tests whether a slower motion is possible. We distinguish between the following cases:

- Initial velocity $v_i = 0$ - In this case, the algorithm checks the maximum waiting duration possible on the current node. It generates a motion in which we hold for a duration $t_{exceed} - t_{trav}$ and then accelerates to arrive exactly on the boundary of the USI.
- Edge length $l / v_i \geq t_{exceed}$ - When the edge length divided by the initial velocity is larger than the traversal time to exceed, this means the vehicle can at least hold its initial velocity. In this case, the algorithm sets the traversal time equal to t_{exceed} , and tries to generate a motion that arrives at the end of the edge at the highest possible v_f .
- Edge length $l / v_i < t_{exceed}$ - In this case, the vehicle has to decelerate to make exceed the t_{exceed} . If a $v_{limit} < v_i$ was defined, the algorithm first checks if it can generate a motion exceeding t_{exceed} while still reaching the same v_{lim} as in B). This is done by decelerating earlier. If this is not possible, the final velocity is lowered to exceed the t_{exceed} .

A.3. Model Verification

This section shows a selection of the steps that were taken to verify the simulation model.

A.3.1. Route Planning with Kinematics

In Figure A.9, the velocity profile for a departing aircraft without any USI of other vehicles to avoid is shown. Figure A.10 shows how this velocity profile maps to the layout. We can see that the vehicle slows down to a turn velocity in all sharp turns. Furthermore, the vehicle accelerates to the maximum Tug-Enabled Taxiing velocity of 11.8 m/s when on the way to the decoupling location. After decoupling, it reaches the maximum Multi-Engine Taxiing velocity of 15 m/s and departs from runway 36L. In Figure A.9 we also observe waiting time to start the aircraft engines and waiting time to decouple the tug from the aircraft.

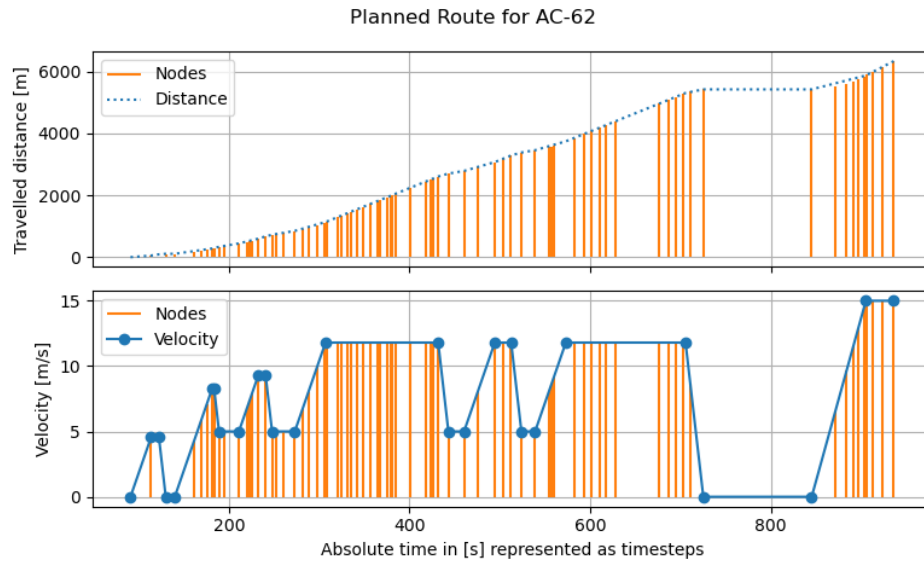


Figure A.9: Velocity profile for towed aircraft that pushes back, taxis, decouples and then departs from the airport.

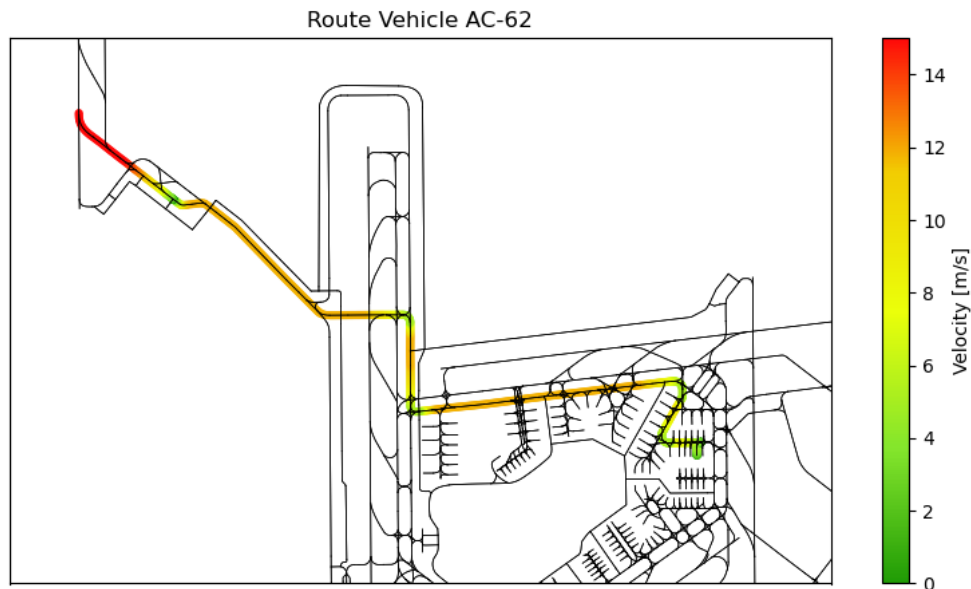


Figure A.10: Velocity heatmap for towed aircraft that pushes back, taxis, decouples and then departs.

A.3.2. Live Animation of Simulation Model

PyGame was used to develop a live animation of the simulation model with playback functionality. This visual tool was used for continuous verification. Single agents were tracked throughout the system and the interaction between multiple agents in conflicts was analyzed. [Figure A.11](#) shows the center taxiways, [Figure A.12](#) shows an example bay area, and [Figure A.13](#) shows an example runway and decoupling area.

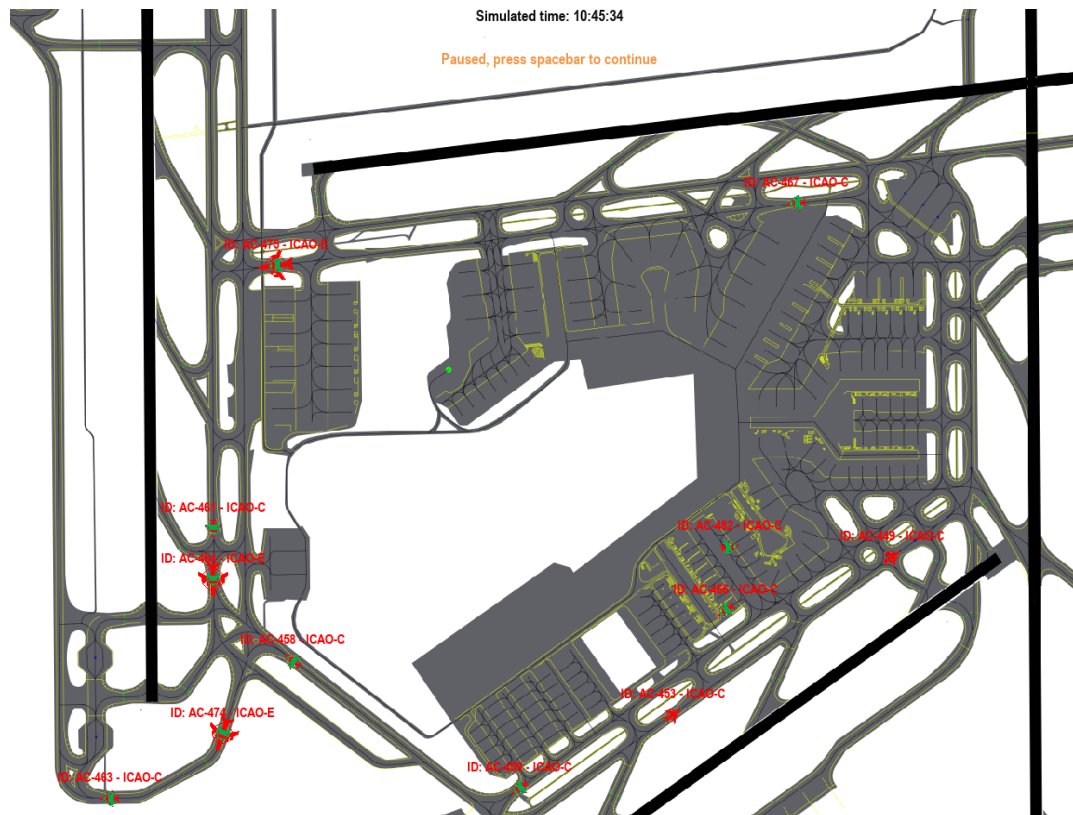


Figure A.11: Screenshot of the central taxiway area in the model of Amsterdam Airport Schiphol.

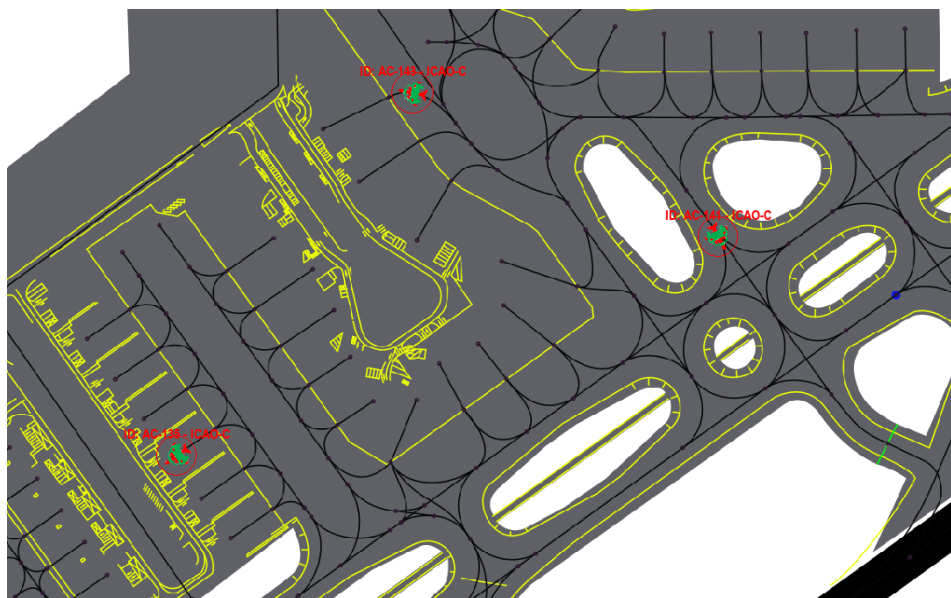


Figure A.12: Screenshot of one of the gate areas (C/D-piers) in the model of Amsterdam Airport Schiphol.

B

Simulation Setup

This section presents the main parameters used in the simulation. The algorithm input parameters are presented first in [section B.1](#). Second, we show the vehicle parameters such as kinematic parameters, engine start times, shape sizes, mapping from aircraft to ICAO category, and wake turbulence separation values in [section B.2](#). Finally, a summary of the scheduled flights and runway mode of operation (RMO) is given in [section B.3](#).

B.1. Algorithm Input Parameters

The main parameters used in the planning algorithm are shown in [Table B.1](#). Flights are rescheduled by the Airport Operations Agent every h_{skdl} for the next w_{skdl} minutes. The Routing Agent plans with a rolling window of w_{plng} minutes and replans every h_{plng} minutes, or when rescheduling took place. The Localized Agents instruct Moving Agents for the next w_{exec} and reinstruct them every h_{exec} , or when the Routing Agent changes the plan. The safety distance D is added to the combined shape of two vehicles, as was discussed in the main paper.

Table B.1: Algorithm Parameters used for simulations.

Name	Value	Description
w_{skdl}	60 minutes	flight scheduling window
h_{skdl}	20 minutes	flight rescheduling period
w_{plng}	30 minutes	planning window
h_{plng}	15 minutes	replanning period
w_{exec}	10 minutes	execution instruction window
h_{exec}	1 minute	re-instruction period
D	30 meters	additional safety distance added to the combined shape of two vehicles

B.2. Vehicle Input Parameters

As mentioned in the paper, each aircraft was mapped to one of the six ICAO categories. Kinematic parameters used for each vehicle type are reported in [subsection B.2.1](#). Engine start times are discussed in [subsection B.2.2](#). The definition of the shape for each category are given in [subsection B.2.3](#). Finally, the parameters to define the minimum separation between two departures is shown in [subsection B.2.4](#).

B.2.1. Kinematic Parameters

The movements of the vehicles are constrained by the maximum velocity v_{max} , acceleration rate acc and deceleration rate dec as shown in [Table B.2](#). Additionally, a maximum turn velocity v_{max} is defined for turns that have a radius of curvature smaller than r_{turn} . Feasible departure runway entries are selected based on the minimum take-off distance d_{TO} . Values were obtained from estimations, that are based on interviews with operational experts and on TaxiBot specifications.

Table B.2: Kinematic parameters per category. *sType*: simulated type, AC+TB: aircraft and tug combination, TMO: taxi mode of operation, MET: Multi-Engine Taxiing, TB: Tug-Enabled Taxiing, v_{max} : maximum velocity, acc : acceleration rate, dec : deceleration rate, r_{turn} : maximum turn radius, v_{turn} : maximum turn velocity, d_{TO} : minimum take-off distance.

<i>sType</i>	<i>TMO</i>	$v_{max}[m/s]$	$acc[m/s^2]$	$dec[m/s^2]$	$r_{turn}[m/s]$	$v_{turn}[m/s]$	$d_{TO}[m]$
ICAO-A	MET	15	0.25	-0.75	100	5	1200
ICAO-B	MET	15	0.25	-0.75	100	5	1700
ICAO-C	MET	15	0.25	-0.75	100	5	2500
ICAO-D	MET	15	0.25	-0.75	100	5	2500
ICAO-E	MET	15	0.25	-0.75	100	5	3100
ICAO-F	MET	15	0.25	-0.75	100	5	3100
AC + TB	TB	11.8	0.20	-0.60	100	5	N/A

B.2.2. Engine Start Parameters

The duration of engine start per simulated aircraft type category and per scenario are shown in [subsection B.2.2](#).

Table B.3: Engine Start + Warm-Up times per *sType* category in seconds. *sType*: simulated type, ES: engine start.

<i>sType/Scenario</i>	<i>Short ES</i>	<i>Medium ES</i>	<i>Long ES</i>
<i>ICAO-A</i>	135	180	225
<i>ICAO-B</i>	135	180	225
<i>ICAO-C</i>	135	180	225
<i>ICAO-D</i>	270	360	450
<i>ICAO-E</i>	270	360	450
<i>ICAO-F</i>	270	360	450

B.2.3. Shape Parameters

Dimensions that define the shape of the aircraft in our simulations are the length L , width W , and offset radius r_{offset} , as shown in [Table B.4](#). The shape of each aircraft type category is calculated with [Equation B.1](#). During Tug-Enabled Taxiing, the aircraft-tug combination is assumed to inherit the shape of the aircraft.

$$r = \frac{\max(\text{Length}, \text{Width})}{2} + r_{offset} \quad (\text{B.1})$$

Table B.4: Length, Width, radius offset (r_{offset}) and radius (r) per simulated ICAO type (*sType*).

<i>sType</i>	<i>Length [m]</i>	<i>Width [m]</i>	<i>r_{offset} [m]</i>	<i>r [m]</i>
<i>ICAO-A</i>	10	12	5	11
<i>ICAO-B</i>	25	22	5	17.5
<i>ICAO-C</i>	40	36	5	25
<i>ICAO-D</i>	54	48	5	32
<i>ICAO-E</i>	72	65	5	41
<i>ICAO-F</i>	73	80	5	45

B.2.4. Wake Turbulence Separation (WTC) Parameters

To ensure sufficient separation between consecutive departures, a time-based separation minimum is set in the simulation. The times are reported in [Table B.5](#) and are dependent on the combination of leader and follower aircraft. The values are based on the RECAT-EU time-based separation minima [62] and were mapped to match the type categories in our simulation.

Table B.5: Minimum Wake Turbulence Separation times in seconds. Based on RECAT-EU time-based separation minima for departure [62] and mapped to corresponding simulated aircraft types used in the simulation. Note that ICAO uses different names for the type categories than we do in our simulation.

Leader/Follower	ICAO-A	ICAO-B	ICAO-C	ICAO-D	ICAO-E	ICAO-F
ICAO-A	80	60	60	60	60	60
ICAO-B	100	60	60	60	60	60
ICAO-C	120	60	60	60	60	60
ICAO-D	120	100	80	60	60	60
ICAO-E	140	120	100	60	60	60
ICAO-F	180	160	140	120	100	60

B.2.5. Mapping to Simulated Aircraft Types

The flights schedule contained an aircraft type for each flight. To cluster the types, each flight was mapped to one of the six simulated ICAO types based on the aircraft name. This mapping is shown in Table B.6.

Table B.6: Mapping from Aircraft Types to ICAO sType used in the simulations. Mapping based on ICAO reference codes [40].

Manufacturer	Name	ICAO sType
Cirrus	Cirrus Vision SF50	ICAO-A
Bombardier	Learjet all types (largest aircraft)	ICAO-B
Embraer	EMB-120 Brasilia Adv.	ICAO-B
Canadair	Canadair CRJ-100/200ER	ICAO-B
Cessna	All types (largest dimensions)	ICAO-B
Dassault	Falcon all types (largest aircraft)	ICAO-C
Bombardier	Q400	ICAO-C
Embraer	Embraer 170/190	ICAO-C
Boeing	B737-300	ICAO-C
Gulfstream	All types (largest aircraft)	ICAO-C
Boeing	B737-300 + winglets	ICAO-C
Airbus	A318-100 (largest 318)	ICAO-C
Boeing	B737-900	ICAO-C
Boeing	B737-600	ICAO-C
Boeing	B737-700	ICAO-C
McDonnell Douglas	McDonnell Douglas MD-80	ICAO-C
Boeing	B737-800	ICAO-C
Bombardier	CS-300 Bombardier	ICAO-C
Boeing	B737-800 NG Winglets	ICAO-C
Airbus	A320-200 Sharklets (largest 320)	ICAO-C
Boeing	B737-700 NG Winglets	ICAO-C
Airbus	A319-100/LR/CJ (largest 319)	ICAO-C
Airbus	A321-200 (largest 321)	ICAO-C
Airbus	A320-200	ICAO-C
Airbus	A310-200/300 Freighter	ICAO-D
Airbus	A300F Freighter	ICAO-D
Airbus	A300F Freighter	ICAO-D
Boeing	B767-300	ICAO-D
Airbus	A330-300	ICAO-E
Airbus	A330-200	ICAO-E
Airbus	A340-300E	ICAO-E
Boeing	B747-400 + 400ER	ICAO-E
Boeing	B747-400 Combi	ICAO-E
Airbus	A350-900	ICAO-E
Airbus	A350-1000	ICAO-E
Boeing	B777F	ICAO-E
Boeing	B747-800F	ICAO-F
Airbus	A380	ICAO-F
Boeing	B777	ICAO-E
Boeing	B787	ICAO-E

B.3. Schedule Input Parameters

A moving average of the spawn rate for departure and arrival aircraft per 15 minutes is visualized in [Figure B.1](#) for the first simulated day and in [Figure B.2](#) for the second day of simulation. In [Table B.7](#), the runway schedule for each days is presented. Finally, [Table B.8](#) shows the number of departure flights per simulated type group (ICAO A-C and ICAO E-F), and the number of flights that uses a push-back, push-pull, or taxi-out maneuver to leave the gate.

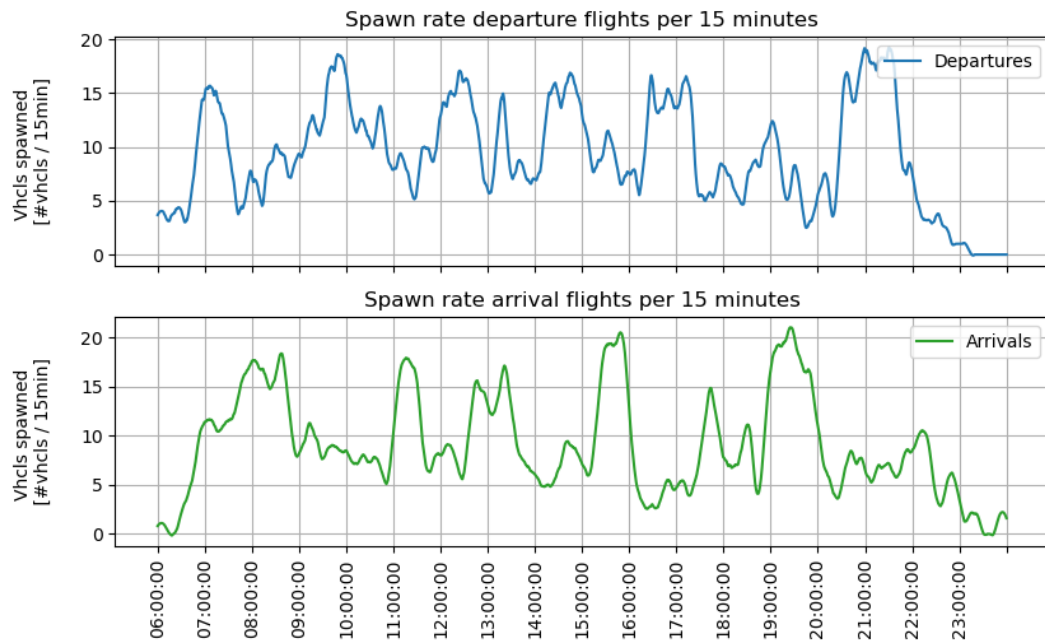


Figure B.1: RMO North: Moving average of the spawn rate per 15 minutes for departure flights (top) and arrival flights (bottom).

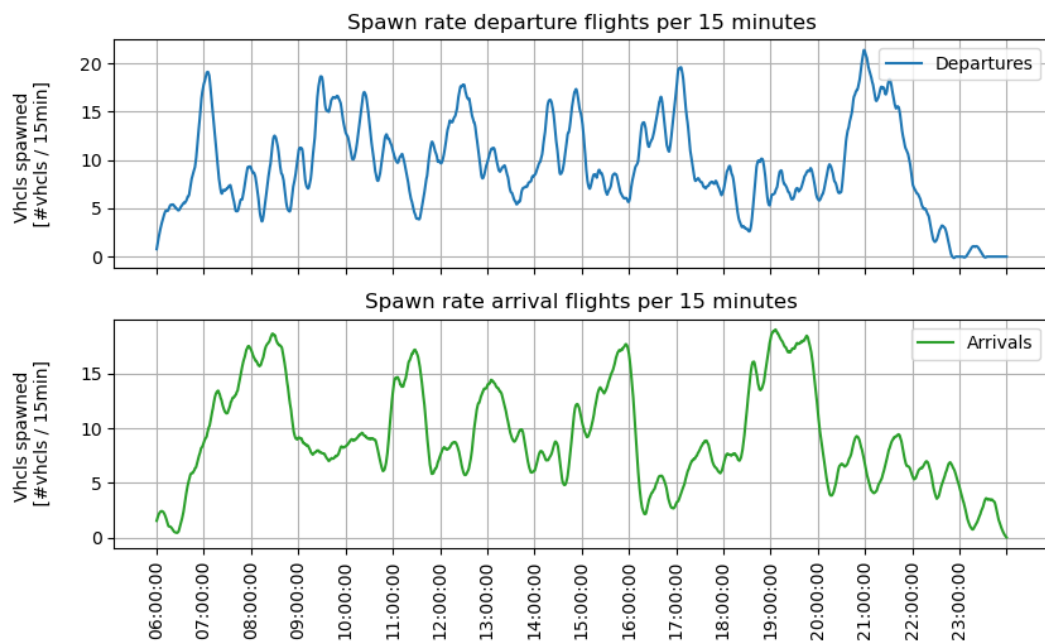


Figure B.2: RMO South: Moving average of the spawn rate per 15 minutes for departure flights (top) and arrival flights (bottom).

Table B.7: Runway Schedule: Activation time of each (set of) runway combinations on both simulated days of operation.

Simulated Day	Activation time	Take-Off Runways	Landing Runways
Day 1: RMO North	0:00	36L	06
Day 1: RMO North	6:45	36L, 36C	06, 36R
Day 1: RMO North	7:45	36L	06, 36R
Day 1: RMO North	9:15	36L, 36C	06, 36R
Day 1: RMO North	11:15	36L	06, 36R
Day 1: RMO North	11:45	36L, 36C	06, 36R
Day 1: RMO North	14:00	36L	06, 36R
Day 1: RMO North	14:15	36L, 36C	06, 36R
Day 1: RMO North	15:30	36L	06, 36R
Day 1: RMO North	16:15	36L, 36C	06, 36R
Day 1: RMO North	17:45	36L	06, 36R
Day 1: RMO North	20:30	36L, 36C	06, 36R
Day 1: RMO North	22:00	36L	06, 36R
Day 2: RMO South	0:30	36C	18R
Day 2: RMO South	1:50	24	18R
Day 2: RMO South	6:55	24, 18L	18R, 18C
Day 2: RMO South	7:45	24	18R, 18C
Day 2: RMO South	9:10	24, 18L	18R, 18C
Day 2: RMO South	9:35	24, 18L	18R
Day 2: RMO South	10:10	24, 18L	18R, 18C
Day 2: RMO South	11:45	24, 18L	18R
Day 2: RMO South	12:35	24, 18L	18R, 18C
Day 2: RMO South	14:50	24, 18L	18R
Day 2: RMO South	13:45	24, 18L	18R, 18C
Day 2: RMO South	15:40	24	18R, 18C
Day 2: RMO South	16:00	24, 18L	18R, 18C
Day 2: RMO South	16:15	24, 18L	18R
Day 2: RMO South	17:45	24	18R
Day 2: RMO South	18:25	24	18R, 18C
Day 2: RMO South	20:30	24, 18L	18R
Day 2: RMO South	22:10	24	18R

Table B.8: Number of departure flights that use push-back, push-pull, and taxi-out. Split per simulated type group. RMO: Runway Mode of Operation.

Simulated Day	sType	Push-back	Push-pull	Taxi-out
Day 1: RMO North	ICAO A-C	479	25	63
	ICAO D-F	59	49	
Day 2: RMO North	ICAO A-C	489	24	58
	ICAO D-F	58	43	

C

Simulation Results

This section presents all results of the simulations performed as part of this MSc thesis. These are partly an elaboration of what has been discussed in the paper and partly results that were not part of the main paper. [section C.1](#) shows the distribution of decoupling standstill time tt_{DC} . In [section C.2](#), we show heatmaps with differences in edge traversal time. Heatmaps with a count of traversals are shown in [section C.3](#). Insights in differences between Tug-Enabled Taxiing and Multi-Engine taxiing per ICAO group are shown in [section C.4](#). Finally, plots of the average variation in taxi time under longer and shorter decoupling are presented in [section C.5](#).

C.1. Distributions Decoupling Standstill Times tt_{DC}

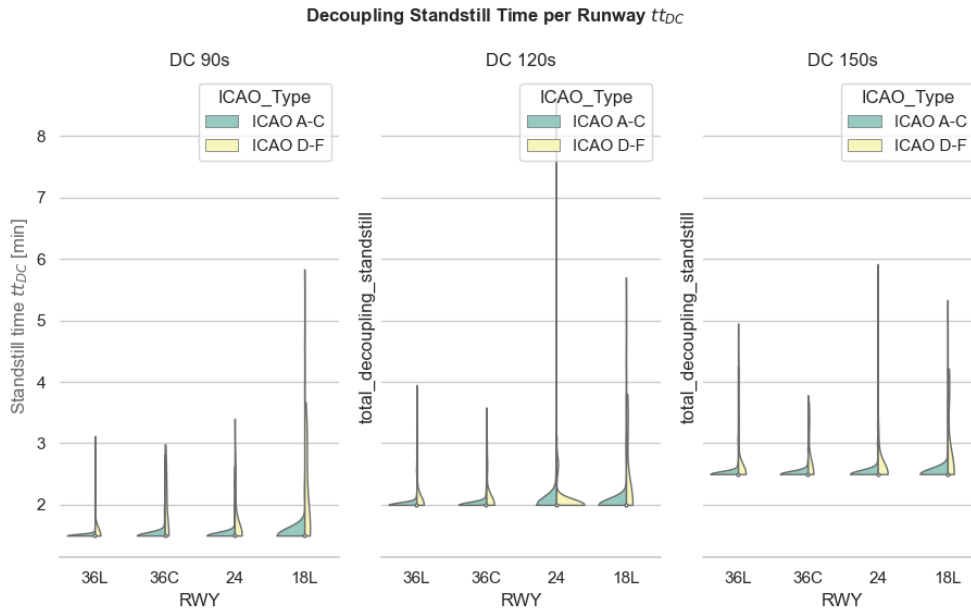


Figure C.1: Decoupling standstill times under medium engine start duration.

C.2. Heatmaps with Differences in Traversal Time per Edge

The heatmaps below show that increase in average traversal time has a peak at decoupling locations. Besides that, there is a consistent increase in taxi times on the taxi routes towards the runways, supporting the idea that the traversal times are generally longer due to the lower taxiing velocities with Tug-Enabled taxiing with respect to MET. Finally, [Figure C.6](#) shows that there are especially large reductions in traversal time for edges at the end of a push-pull path.

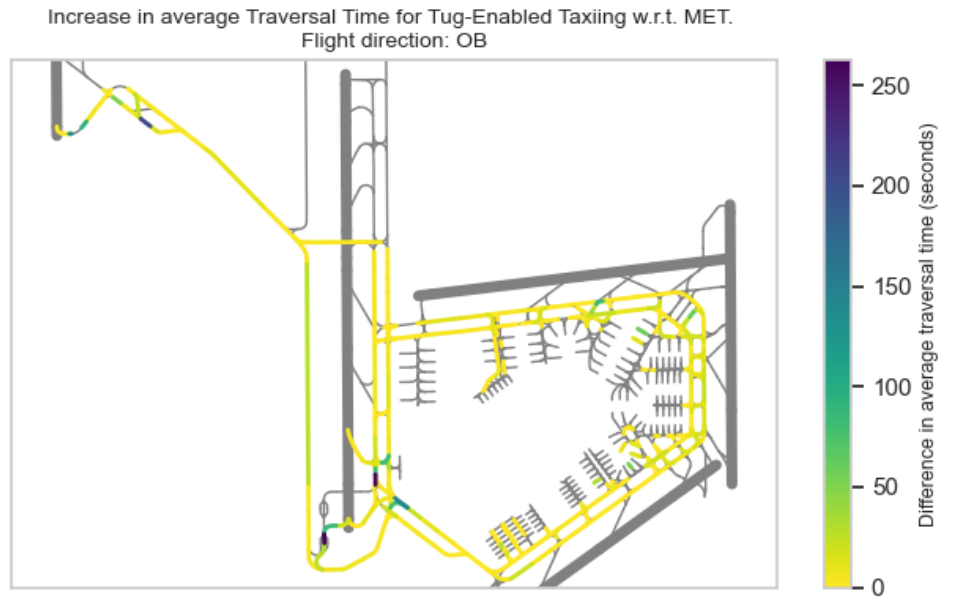


Figure C.2: RMO North - taxi time increase per edge

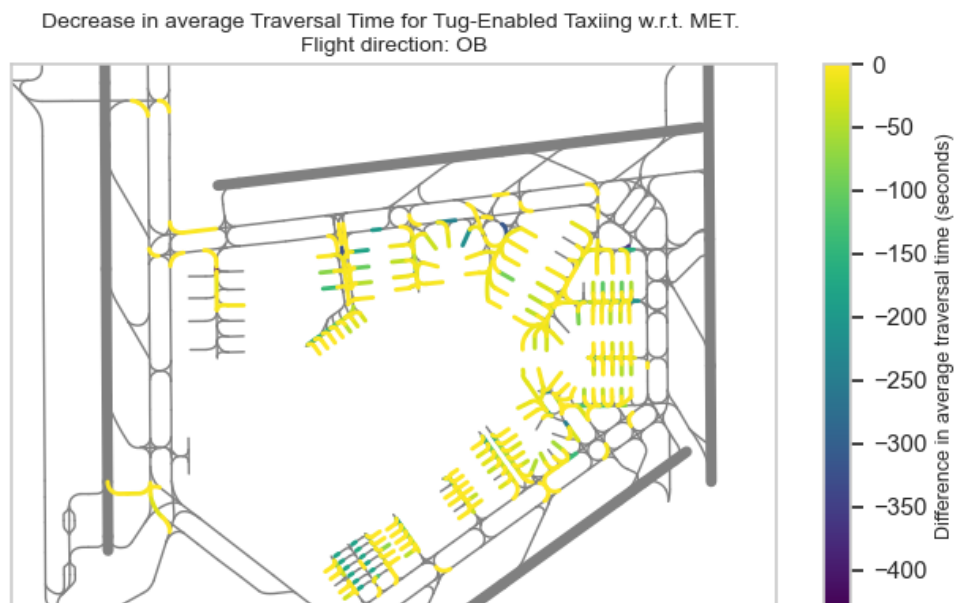


Figure C.3: RMO North - taxi time decrease per edge.

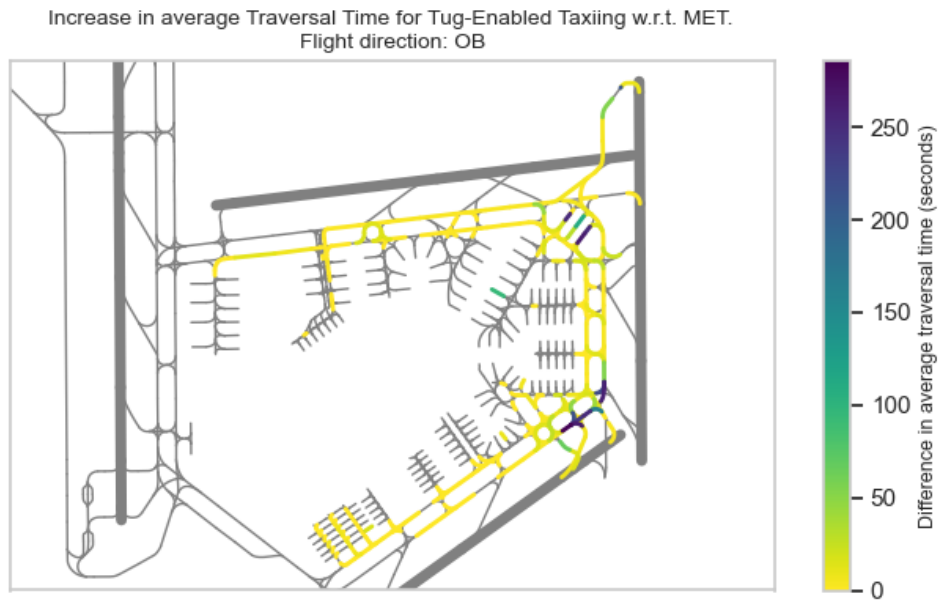


Figure C.4: RMO South - taxi time increase per edge

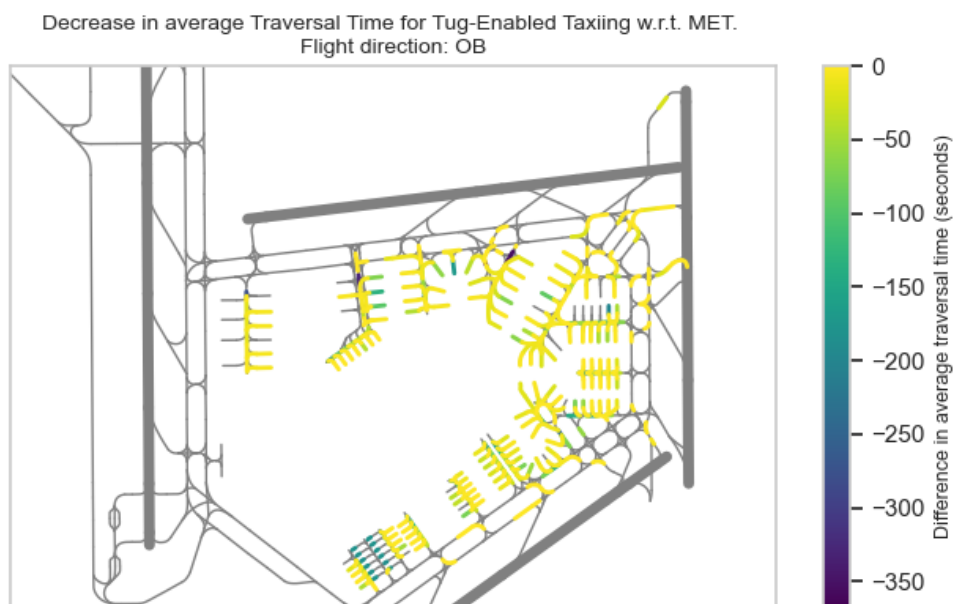


Figure C.5: RMO South - taxi time decrease per edge

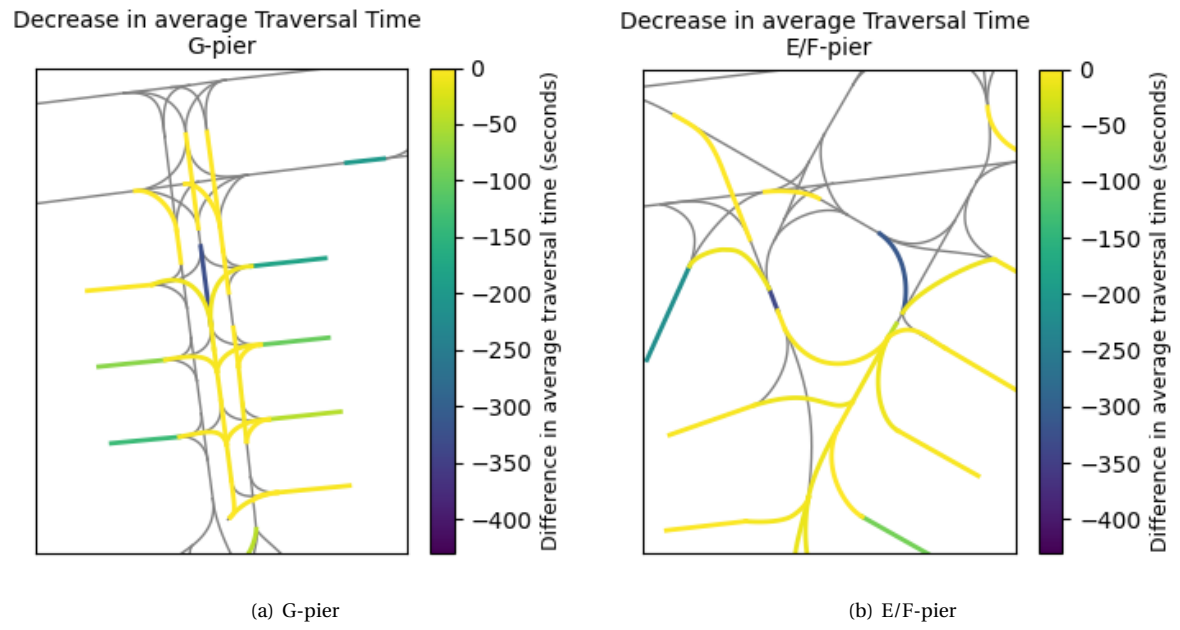
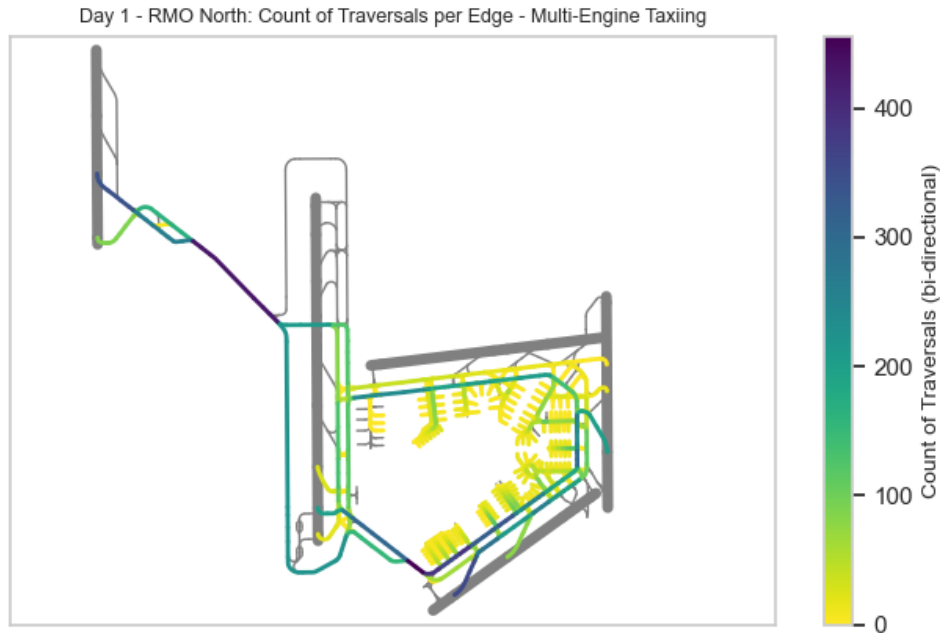


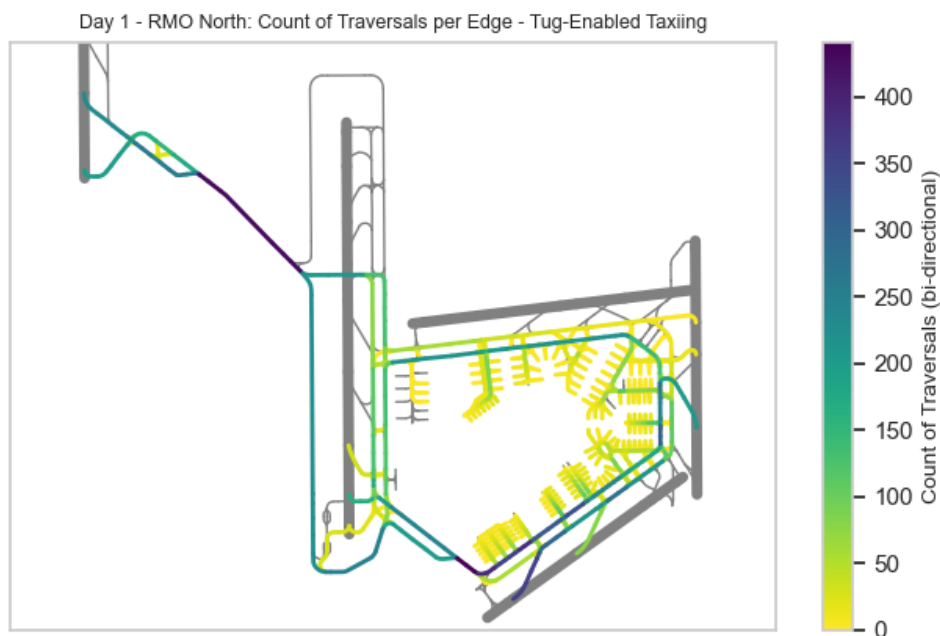
Figure C.6: Decrease in traversal time for bay areas with many push-pull maneuvers.

C.3. Heatmaps with Count of Edge Traversals per Day

Figure C.7 shows the count of traversals per edge on the first simulated day for both Multi-Engine Taxiing and Tug-Enabled Taxiing. The main paper already showed that there are no large differences in taxi distances for departure flights. This is further supported by the lack of large differences between C.7(a) and C.7(b). A similar pattern is visible for day 2, as shown in Figure C.8.

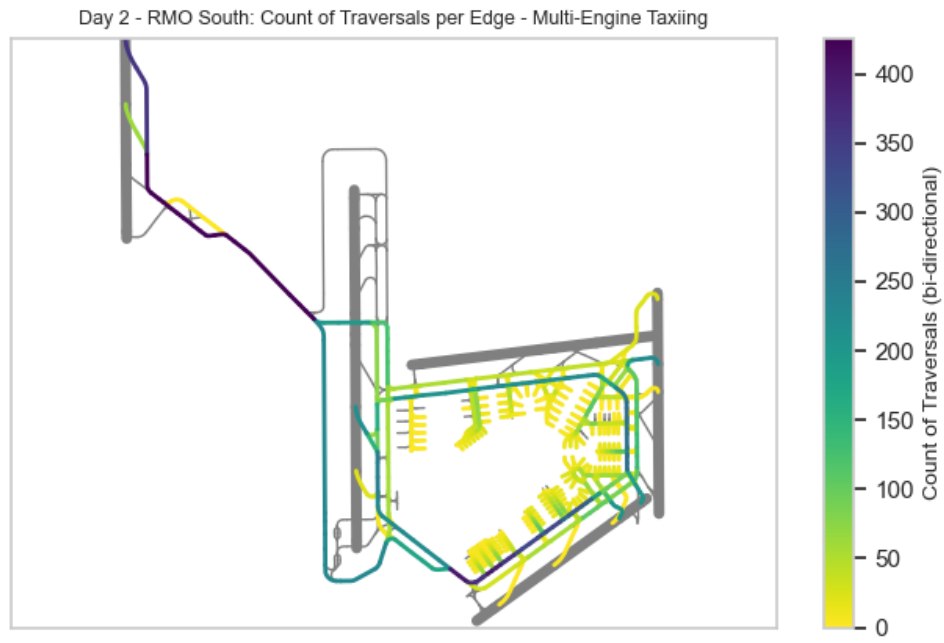


(a) Multi-Engine Taxiing

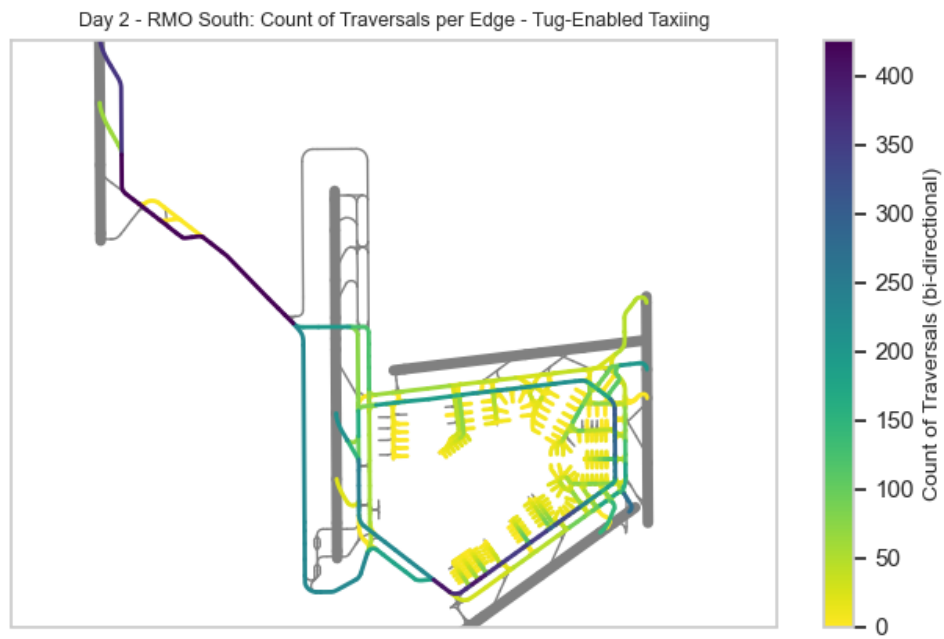


(b) Tug-Enabled Taxiing

Figure C.7: Count of Traversals for Multi-Engine Taxiing and Tug-Enabled taxiing under RMO North.



(a) Multi-Engine Taxiing



(b) Tug-Enabled Taxiing

Figure C.8: Count of Traversals for Multi-Engine Taxiing and Tug-Enabled taxiing under RMO South.

C.4. Analysis of Taxi Time Differences per Aircraft Type Group

As an additional analysis, we analyzed the taxi time differences δtt_{dep} and bay area taxi time differences δtt_{BA} between the group of smaller aircraft (ICAO A-C) and the group of larger aircraft (ICAO D-F). The Mann-Whitney U test was used, as two unpaired groups are compared and the data is not normally distributed and showed significant differences ($p < 0.001$) in both δtt_{dep} and δtt_{BA} for all engine start durations. Analysis was performed for all flights that depart on day 1 and day 2. Results are shown below in Table C.1 and Figure C.9. It can be observed that differences for larger ICAO types are more apparent than for smaller ICAO types.

Table C.1: Mean difference and median difference in total outbound taxi time tt_{dep} and taxi time on bay area tt_{BA} with respect to Multi-Engine Taxiing for Tug-Enabled Taxiing with a decoupling time of 2:00 minutes

		δtt_{dep} w.r.t. MET		δtt_{BA} w.r.t. MET	
		mean [min]	median [min]	mean [min]	median [min]
Short ES	ICAO A-C	2:44	2:37	-1:33	-1:28
	ICAO D-F	0:31	0:40	-4:02	-3:35
Medium ES	ICAO A-C	1:58	1:52	-2:20	-2:13
	ICAO D-F	-0:52	-0:54	-5:25	-5:03
Long ES	ICAO A-C	1:13	1:05	-3:04	-2:58
	ICAO D-F	-2:14	-2:14	-6:58	-6:34

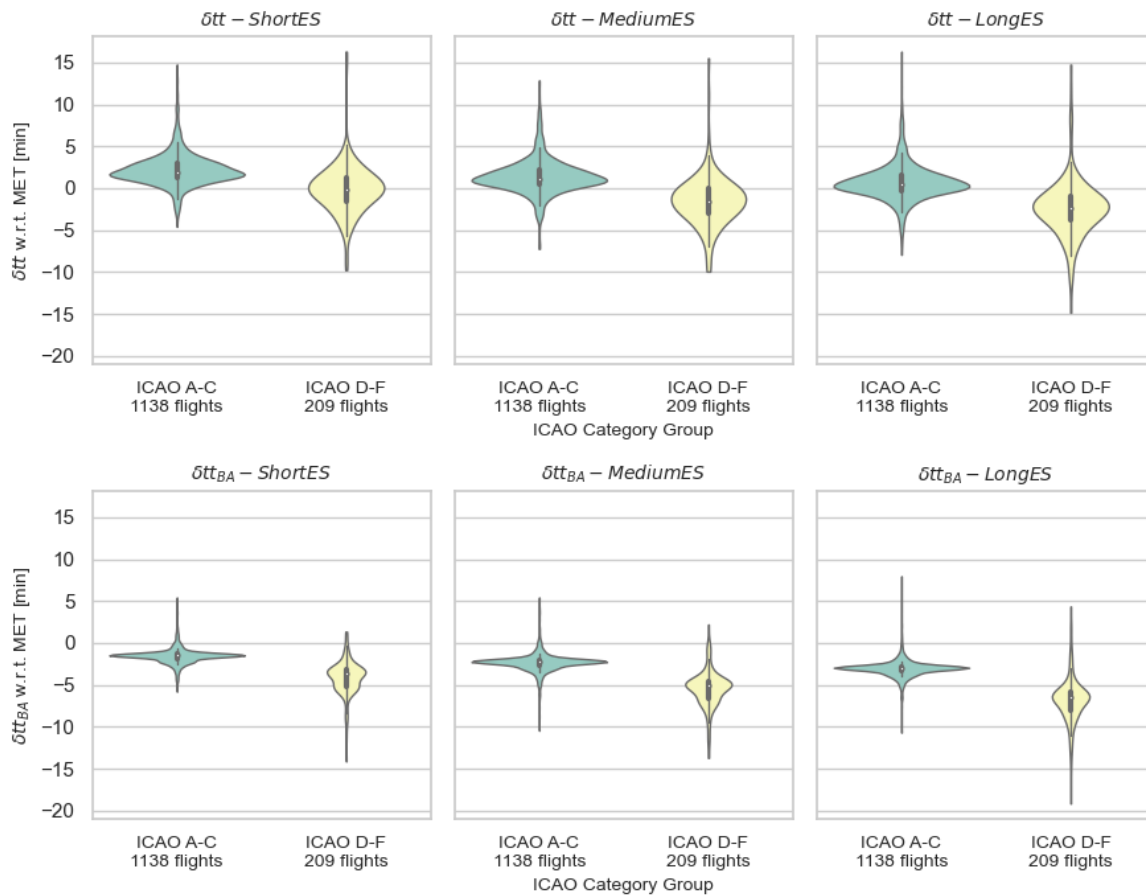


Figure C.9: Distribution of delta taxi times. Total outbound taxi time (top) and taxi time bay area (bottom) for different engine start durations.

C.5. Average Taxi Time Variation for Shorter/Longer Decoupling

The average additional taxi time for longer and shorter decoupling scenarios under RMO North is shown in Figure C.10. A noticeable peak occurs for a decoupling time of 2:30 minutes at 21:30. Visual comparison of the traffic during this peak, presented in Figure C.11, shows that there is additional queue forming for the longer decoupling duration in this scenario that contributes to this increase in taxi time.

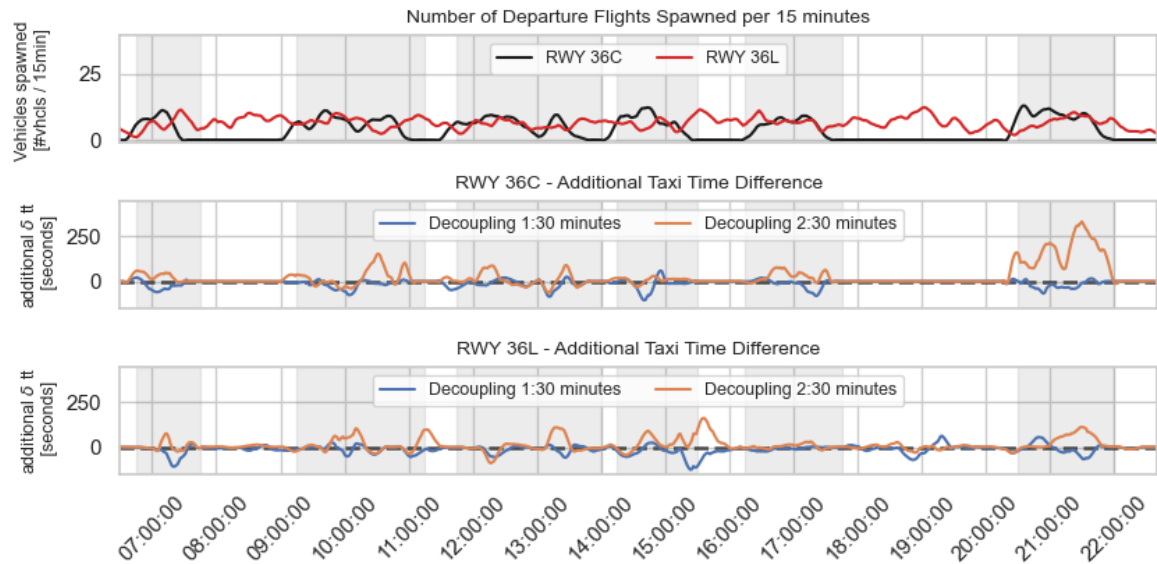


Figure C.10: Additional Taxi Time Differences for Departure Flights under variation of decoupling duration for RMO North.

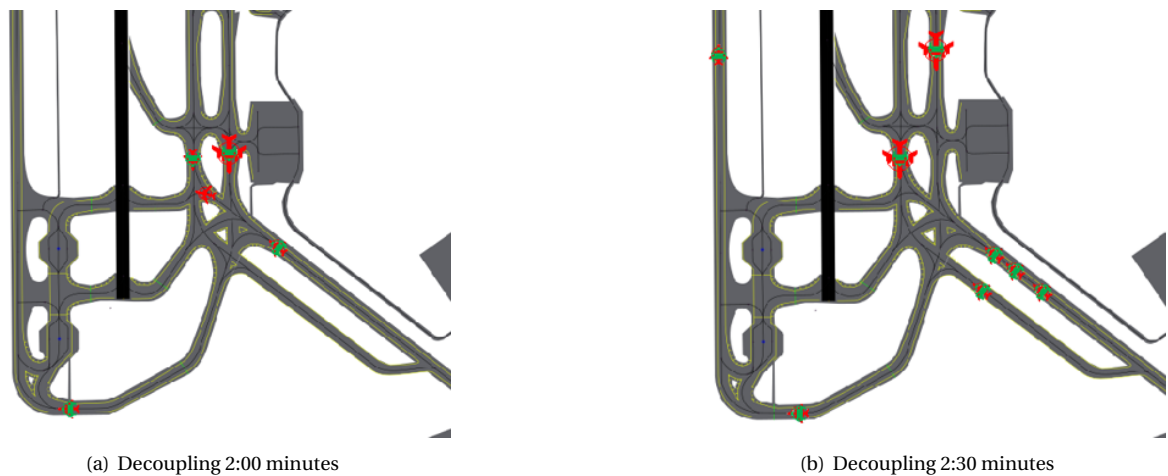


Figure C.11: Queue forming in scenario with longer decoupling duration in the big peak around 21:30h.

For RMO South, Figure C.12 shows the taxi time difference for shorter and longer decoupling as a moving average of 15 minutes over the day. It can be seen that the increases follow a peak-shaped pattern. However, those are not always clearly related to the peaks in the departure schedule. For RMO south, we can therefore not conclude that this peak behavior only happens in the departure peaks.

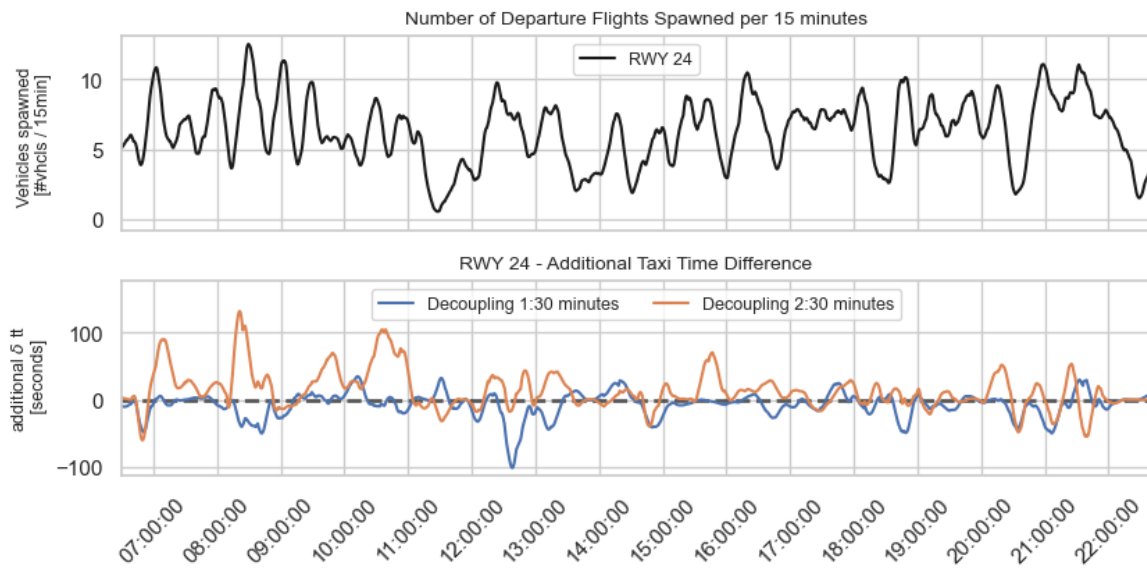


Figure C.12: Additional Taxi Time Differences for Departure Flights under variation of decoupling duration for RMO South.

D

Statistical Elaboration on Experiments

In this chapter, the methods use to compare the KPIs are elaborated. Furthermore, results from statistical tests that were not reported in the main paper are presented.

D.1. Methods Used

Hypotheses testing was performed with several statistical tests. The Shapiro-Wilk test was used to show that the data was not normally distributed. This indicated that we have to use non-parametric tests. For comparisons between the same agents in different scenarios, we therefore used the non-parametric Wilcoxon Signed-Rank tests for paired populations. Comparisons between different groups of agents were performed with the non-parametric Mann-Whitney U test. A significance level of $\alpha = 0.05$ was used, indicating the probability of incorrectly rejecting the true null hypothesis. To correct for family wise error rates in multiple comparisons, the Bonferroni correction was used. With this method, the alpha level is corrected to $\alpha_{bf} = \frac{\alpha}{n}$, where n is the number of tests.

In case a significant difference between two groups is found, we used the Vargha-Delaney A-test [100] as a measure of effect size. This test reports a value between 0 and 1 which is the probability that a randomly selected observation from one group is larger than a randomly selected observation from the other group. A value of 0.50 indicates no difference between the groups. Values above 0.56, 0.64, and 0.71 indicate a small, medium, and large difference, respectively. The same intervals apply below 0.5.

D.2. Comparisons under Variation of Engine Start Times

A) Comparison of taxi time difference δtt and bay area taxi time difference δtt_{BA} under variation of engine start times

Table D.1 shows the results of the comparison of differences in taxi time (δtt_{dep}) between the Tug-Enabled Taxiing Scenarios and the MET scenarios under variation of the engine start duration. Furthermore, Table D.2 presents the same measures for the difference in bay area taxi time (δtt_{BA}).

Table D.1: Comparison of δtt_{dep} for shorter engine start times (-25%) and longer engine start times (+25%) with respect to a medium engine start duration for each decoupling duration. Results for runway 36L (a), 36C (b), 18L (c), and 24 (d). DC: decoupling, ES: engine start, RWY: Runway, S: small effect size, M: medium effect size. Significance level $0.05/2=0.025$

(a) RWY 36L					(b) RWY 36C				
	ES Shorter (-25%)		ES Longer (+25%)			ES Shorter (-25%)		ES Longer (+25%)	
	p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test
DC 1:30 min	<0.001	(0.644, M)	<0.001	(0.37, S)	DC 1:30 min	<0.001	(0.65, M)	<0.001	(0.38, S)
DC 2:00 min	<0.001	(0.643, M)	<0.001	(0.36, S)	DC 2:00 min	<0.001	(0.62, S)	<0.001	(0.39, S)
DC 2:30 min	<0.001	(0.641, M)	<0.001	(0.37, S)	DC 2:30 min	<0.001	(0.60, S)	<0.001	(0.41, S)

(c) RWY 18L					(d) RWY 24				
	ES Shorter (-25%)		ES Longer (+25%)			ES Shorter (-25%)		ES Longer (+25%)	
	p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test
DC 1:30 min	<0.001	(0.65, S)	<0.001	(0.35, M)	DC 1:30 min	<0.001	(0.68, M)	<0.001	(0.35, M)
DC 2:00 min	<0.001	(0.66, S)	<0.001	(0.37, S)	DC 2:00 min	<0.001	(0.67, M)	<0.001	(0.38, S)
DC 2:30 min	<0.001	(0.642, M)	<0.001	(0.37, S)	DC 2:30 min	<0.001	(0.645, M)	<0.001	(0.34, M)

Table D.2: Comparison of δtt_{BA} for shorter engine start times (-25%) and longer engine start times (+25%) with respect to a medium engine start duration for each decoupling duration. Results for runway 36L (a), 36C (b), 18L (c), and 24 (d). DC: decoupling, ES: engine start, RWY: Runway, S: small effect size, M: medium effect size. Significance level $0.05/2=0.025$

(a) RWY 36L					(b) RWY 36C				
	ES Shorter (-25%)		ES Longer (+25%)			ES Shorter (-25%)		ES Longer (+25%)	
	p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test
DC 1:30 min	<0.001	(0.73, L)	<0.001	(0.28, L)	DC 1:30 min	<0.001	(0.74, L)	<0.001	(0.285, L)
DC 2:00 min	<0.001	(0.723, M)	<0.001	(0.28, L)	DC 2:00 min	<0.001	(0.74, L)	<0.001	(0.28, L)
DC 2:30 min	<0.001	(0.73, L)	<0.001	(0.28, L)	DC 2:30 min	<0.001	(0.73, L)	<0.001	(0.293, M)

(c) RWY 18L					(d) RWY 24				
	ES Shorter (-25%)		ES Longer (+25%)			ES Shorter (-25%)		ES Longer (+25%)	
	p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test
DC 1:30 min	<0.001	(0.69, M)	<0.001	(0.32, M)	DC 1:30 min	<0.001	(0.719, M)	<0.001	(0.28, L)
DC 2:00 min	<0.001	(0.71, M)	<0.001	(0.32, M)	DC 2:00 min	<0.001	(0.73, L)	<0.001	(0.28, L)
DC 2:30 min	<0.001	(0.71, M)	<0.001	(0.31, M)	DC 2:30 min	<0.001	(0.73, L)	<0.001	(0.28, L)

B) Comparison of absolute taxi times and bay area taxi times in Tug-Enabled Taxiing scenarios under variation of engine start times

Within the nine Tug-Enabled taxiing scenarios on each day, we compared the taxi times for departure aircraft tt_{dep} to each other for different engine start durations. The scenario with a medium engine start duration was chosen as base case. Results are shown in Table D.3 for the RMO North runways and in Table D.4 for the RMO South runways. The p-value from the Wilcoxon Signed Rank Test is always reported, whereas the A-test was only performed when there was a significant difference between the groups. A significance level of $\alpha = 0.05/2 = 0.0025$ was used, as we compare to the base scenario of medium engine start twice.

Table D.3: Comparison of tt_{dep} for shorter engine start times (-25%) and longer engine start times (+25%) with respect to a medium engine start duration for each decoupling duration. Results for runway 36L (a) and 36C (b). DC: decoupling, ES: engine start, RWY: Runway. Significance level $0.05/2=0.025$

(a) RWY 36L					(b) RWY 36C				
	ES Shorter (-25%)		ES Longer (+25%)			ES Shorter (-25%)		ES Longer (+25%)	
	p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test
DC 1:30 min	0.98	-	0.60	-	DC 1:30 min	0.72	-	0.03	-
DC 2:00 min	0.71	-	0.58	-	DC 2:00 min	0.85	-	0.87	-
DC 2:30 min	0.54	-	0.43	-	DC 2:30 min	0.32	-	0.88	-

Table D.4: Comparison of tt_{dep} for shorter engine start times (-25%) and longer engine start times (+25%) with respect to a medium engine start duration for each decoupling duration. Results for runway 18L (a) and 24 (b). DC: decoupling, ES: engine start, RWY: Runway. Significance level $0.05/2=0.025$

(a) RWY 18L					(b) RWY 24				
	ES Shorter (-25%)		ES Longer (+25%)			ES Shorter (-25%)		ES Longer (+25%)	
	p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test
DC 1:30 min	0.37	-	0.50	-	DC 1:30 min	0.09	-	<0.001	(0.53, N)
DC 2:00 min	0.29	-	0.23	-	DC 2:00 min	0.44	-	<0.001	(0.53, N)
DC 2:30 min	0.04	-	0.07	-	DC 2:30 min	0.96	-	0.60	-

Table D.5: Comparison of tt_{BA} for shorter engine start times (-25%) and longer engine start times (+25%) with respect to a medium engine start duration for each decoupling duration. Results for runway 36L (a), 36C (b), 18L (c) and 24 (d). Zero differences included in ranking. DC: decoupling, ES: engine start, RWY: Runway. Significance level $0.05/2=0.025$

(a) RWY 36L					(b) RWY 36C				
	ES Shorter (-25%)		ES Longer (+25%)			ES Shorter (-25%)		ES Longer (+25%)	
	p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test
DC 1:30 min	0.77	-	0.57	-	DC 1:30 min	0.62	-	0.62	-
DC 2:00 min	0.50	-	1.00	-	DC 2:00 min	0.63	-	1.00	-
DC 2:30 min	0.93	-	1.00	-	DC 2:30 min	0.82	-	0.90	-

(c) RWY 18L					(d) RWY 24				
	ES Shorter (-25%)		ES Longer (+25%)			ES Shorter (-25%)		ES Longer (+25%)	
	p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test
DC 1:30 min	0.73	-	0.27	-	DC 1:30 min	0.21	-	0.34	-
DC 2:00 min	0.91	-	0.91	-	DC 2:00 min	0.64	-	0.14	-
DC 2:30 min	0.91	-	0.97	-	DC 2:30 min	0.64	-	0.72	-

C) Comparison of absolute taxi times in Multi-Engine Taxiing Scenarios under Variation of Engine Start Times
 Table D.6 shows the results from the comparison of MET scenarios under variation of engine start times, as was discussed in the main paper. Taxi times under a longer ES duration and shorter ES duration were compared to taxi times with medium engine start. In this comparison, the engine start time was excluded to analyze to what extent the scenarios differ. Table D.6 shows that there is no difference that is both significant and relevant in size.

Table D.6: Comparison of $tt_{dep} - EStime$ for shorter engine start times (-25%) and longer engine start times (+25%) with respect to a medium engine start duration for Multi-Engine Taxiing scenarios. Results for runway 36L (a), 36C (b), 18L (c), and 24 (d). ES: engine start, RWY: Runway, N: negligible effect size. Significance level $0.05/2=0.025$

(a) RWY 36L					(b) RWY 36C				
	ES Shorter (-25%)		ES Longer (+25%)			ES Shorter (-25%)		ES Longer (+25%)	
	p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test
MET	0.31	-	0.54	-	MET	0.12	-	0.91	-

(c) RWY 18L					(d) RWY 24				
	ES Shorter (-25%)		ES Longer (+25%)			ES Shorter (-25%)		ES Longer (+25%)	
	p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test
MET	0.087	-	0.27	-	MET	0.52	-	0.015	(0.52, N)

D.3. Comparisons under Variation of Decoupling Times

Table D.7 shows the comparison of taxi time difference δtt under variation of decoupling times, whereas Table D.8 shows the difference in bay area taxi time difference δtt_{BA} under the same conditions. As can be seen, δtt_{BA} is not significantly influenced by the variation of decoupling time.

Table D.7: Comparison of δtt_{dep} for shorter decoupling times (-25%) and longer decoupling times (+25%) with respect to a medium decoupling duration (2:00 min) for each engine start duration. Results for runway 36L (a), 36C (b), 18L (c), and 24 (d). DC: decoupling, ES: engine start, RWY: Runway, S: small effect size, M: medium effect size. Significance level $0.05/2=0.025$

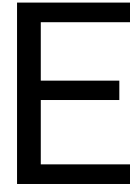
(a) RWY 36L					(b) RWY 36C				
	DC Shorter (-25%)		DC Longer (+25%)			DC Shorter (-25%)		DC Longer (+25%)	
	p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test
ES Short	<0.001	(0.37, S)	<0.001	(0.61, S)	ES Short	<0.001	(0.37, S)	<0.001	(0.63, S)
ES Medium	<0.001	(0.39, S)	<0.001	(0.60, S)	ES Medium	<0.001	(0.39, S)	<0.001	(0.63, S)
ES Long	<0.001	(0.40, S)	<0.001	(0.61, S)	ES Long	<0.001	(0.41, S)	<0.001	(0.63, S)

(c) RWY 18L					(d) RWY 24				
	DC Shorter (-25%)		DC Longer (+25%)			DC Shorter (-25%)		DC Longer (+25%)	
	p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test
ES Short	<0.001	(0.40, S)	<0.001	(0.60, S)	ES Short	<0.001	(0.35, M)	<0.001	(0.63, S)
ES Medium	<0.001	(0.41, S)	<0.001	(0.61, S)	ES Medium	<0.001	(0.37, S)	<0.001	(0.637, S)
ES Long	<0.001	(0.37, S)	<0.001	(0.60, S)	ES Long	<0.001	(0.37, S)	<0.001	(0.59, S)

Table D.8: Comparison of δtt_{BA} for shorter decoupling times (-25%) and longer decoupling times (+25%) with respect to a medium decoupling duration (2:00 min) for each engine start duration. Results for runway 36L (a), 36C (b), 18L (c), and 24 (d). Zero differences included in ranking. DC: decoupling, ES: engine start, RWY: Runway, N: negligible effect size. Significance level $0.05/2=0.025$

(a) RWY 36L					(b) RWY 36C				
	DC Shorter (-25%)		DC Longer (+25%)			DC Shorter (-25%)		DC Longer (+25%)	
	p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test
ES Short	0.47	-	0.82	-	ES Short	0.21	-	0.27	-
ES Medium	0.16	-	0.57	-	ES Medium	0.43	-	0.14	-
ES Long	0.57	-	0.70	-	ES Long	0.45	-	0.10	-

(c) RWY 18L					(d) RWY 24				
	DC Shorter (-25%)		DC Longer (+25%)			DC Shorter (-25%)		DC Longer (+25%)	
	p-val	A-test	p-val	A-test		p-val	A-test	p-val	A-test
ES Short	0.18	-	0.63	-	ES Short	0.36	-	0.23	-
ES Medium	0.89	-	0.68	-	ES Medium	0.99	-	0.10	-
ES Long	0.58	-	0.84	-	ES Long	0.82	-	0.38	-



Recommendations for Future Work

Recommendations for future research in the topic of engine-off taxiing and hierarchical multi-agent path planning for airport surface movement operations are as follows:

- **Extension of Tug-Enabled Taxiing Operations in Model:** As noted in the main paper, tugs are assumed to drive back to the parking position without any interaction with other traffic. Besides, the movement of the tug from the decoupling position to the all-clear point was not modelled, whereas there is a need to cross taxiways in the current airport layout at AAS. It is therefore recommended to include these operations in order to improve the realism of Tug-Enabled Taxiing Operations at AAS.
- **Tug Assignment:** The inclusion of tug assignment to aircraft, and modelling the movement from the tug parking position to the aircraft ramps is advised in future research in order to evaluate the required tug fleet. Results from this study have provided first insights in which flights benefit from Tug-Enabled Taxiing most and can be used in this task assignment.
- **Evaluate Environmental and Economic Benefits of Tug-Enabled Taxiing:** This study has mainly focused on the taxi times on the global level and in the bay areas. It is recommended to include additional KPIs related to fuel savings and cost in order to evaluate the concept on a broader scale. In future extensions of the model, it could even be interesting to include this in the optimisation objective of the routing algorithm.
- **Addition of airport processes in model:** The modular simulator developed in this project has the potential to be further aligned with current operations. The inclusion of gate occupancy is suggested as a first step. This would allow for modelling the need for inbound holding operations, reflecting inbound aircraft that cannot enter their gate because of an outbound aircraft occupying it. Furthermore, the deployment of Tug-Enabled Taxiing also allows for experiments with new processes, such as outbound holding.
- **Variation of vehicle parameters:** The kinematic parameters of the vehicles in the simulator could be varied to evaluate how kinematics influence the routing. Besides, engine start times can be further diversified in order to match individual cases better.
- **Variation of algorithm parameters:** The performance of the routing algorithm could be evaluated under variation of the input parameters that were discussed in [section B.1](#). Analysis of variation of the planning window w_{plng} and replanning period h_{plng} are suggested first steps.
- **Generalization of results for other airports:** We focused on a case study of AAS in this work. However, the simulation model was set up in a modular way with the idea to allow for the study of other airports. It is recommended to evaluate how results of the comparison between Tug-Enabled Taxiing and Multi-Engine Taxiing change for these airports.
- **Improve constraint setting in special cases:** As mentioned in the main paper, we observed the overlap of shapes of two aircraft in queues with low velocities. This problem is caused by the fact that this

situation is underconstrained in the current implementation. Constraints are set based on the route of a prioritized vehicle, without knowing the kinematics of a potential deprioritized vehicle. Furthermore, we plan for the deprioritized vehicle as a point. Although center-center collisions are prevented in this way, there are situations in which the center of the deprioritized vehicle arrives exactly at the start of a safe interval, whereas its shape already overlaps earlier. A possible solution to this is to be more conservative in planning. One idea is to increase the safety distance D , that is part of the combined shape r_s . This will extend the length of the USI set by the prioritized agent. Another idea is to focus more on local conflict resolution for these special cases, for example by self-separation by Moving Agents or by reasoning by the Localized Agents.

- **Compare simulation model to actual operations:** A comparison to actual operations will show how the concept of hierarchical multi-agent planning compares to conventional operations.

Bibliography

- [1] Baangebruikcijfers 2019. URL https://bezoekbas.nl/wp-content/uploads/2020/02/Baangebruikcijfers_2019.pdf. Accessed: 01-03-2021.
- [2] AEON. Home Page - AEON. URL <https://www.aeon-project.eu/>. Accessed: 16-03-2021.
- [3] Anton Andreychuk. Multi-Agent Path Finding with Kinematic Constraints via Conflict Based Search. In *Artificial Intelligence*, pages 29–45, Cham, 2020. Springer International Publishing. ISBN 978-3-030-59535-7.
- [4] Anton Andreychuk, Konstantin Yakovlev, Dor Atzmon, and Roni Stern. Multi-Agent Pathfinding with Continuous Time. In *IJCAI International Joint Conference on Artificial Intelligence*, volume August, pages 39–45. International Joint Conferences on Artificial Intelligence, 2019. ISBN 9780999241141. doi: 10.24963/ijcai.2019/6.
- [5] Jason A. D. Atkin, Edmund K. Burke, and Stefan Ravizza. The Airport Ground Movement Problem: Past and Current Research and Future Directions. In *4th International Conference on Research in Air Transportation*, pages 131–138, Budapest, 6 2010. ISBN 9781450714686.
- [6] Dor Atzmon, Roni Stern, Ariel Felner, Glenn Wagner, R. Barták, and Neng-Fa Zhou. Robust Multi-Agent Path Finding. *SOCS 2018*, pages 2–9, 2018.
- [7] Paul Baran. On Distributed Communications Networks. *IEEE Transactions on Communications Systems*, 12(1):1–9, 1964. ISSN 00961965. doi: 10.1109/TCOM.1964.1088883.
- [8] Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Pathfinding Problem. In *Proceedings of the Seventh Annual Symposium on Combinatorial Search (SoCS 2014)*, pages 19–27, Prague, Czech Republic, 8 2014.
- [9] Bram Benda. Agent-Based Modelling and Analysis of Non-Autonomous Airport Ground Surface Operations, 2020. Delft University of Technology.
- [10] V.A. Bijsterbosch and W.H. Dalmeijer. Gebruiksprognose 2021, 2020. URL <https://www.schiphol.nl/en/you-and-schiphol/page/gebruiksprognose-alles-wat-je-maar-wilt-weten/>.
- [11] Zahy Bnaya and Ariel Felner. Conflict-Oriented Windowed Hierarchical Cooperative A*. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3743–3748. Institute of Electrical and Electronics Engineers Inc., 9 2014. doi: 10.1109/ICRA.2014.6907401.
- [12] Eric Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America*, 99(SUPPL. 3):7280–7287, 5 2002. ISSN 00278424. doi: 10.1073/pnas.082080899.
- [13] Liron Cohen, Tansel Uras, T. K. Satish Kumar, and Sven Koenig. Optimal and Bounded-Suboptimal Multi-Agent Motion Planning. In *Proceedings of the Twelfth International Symposium on Combinatorial Search (SoCS 2019)*, pages 44–51, Napa, California, USA, 2019. ISBN 9781577358084.
- [14] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. MIT Press, Cambridge, Mass., third edition, 2009.
- [15] M. de Weerd, A. ter Mors, and C. Witteveen. Multi-agent Planning An introduction to planning and coordination. *Handbook of Systemic Autoimmune Diseases*, 2005. URL https://www.researchgate.net/publication/216824952_Multi-agent_Planning. Accessed: 29-03-2021.
- [16] Boris De Wilde, Adriaan W. Ter Mors, and Cees Witteveen. Push and Rotate: A complete Multi-agent Pathfinding algorithm. *Journal of Artificial Intelligence Research*, 51:443–492, 10 2014. ISSN 10769757. doi: 10.1613/jair.4447.

- [17] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 12 1959. ISSN 0029599X. doi: 10.1007/BF01386390.
- [18] Wouter Duchateau. Multi-Agent Pickup-and-Delivery in a Distributed Baggage Handling System, 2021. Delft University of Technology.
- [19] Edmund H. Durfee. Distributed Problem Solving and Planning. In Gerhard Weiss, editor, *MultiAgent Systems: A Modern Approach to Distributed Artificial Intelligence*, chapter 3, pages 118–149. MIT Press, San Francisco, CA, USA, 2001.
- [20] Google Earth. Satellite image of Schiphol. URL <https://earth.google.com/web/>. Accessed: 02-03-2021.
- [21] E. Erdem, D. Kisa, U. Öztok, and Peter Schüller. A General Formal Framework for Pathfinding Problems with Multiple Agents. *AAAI*, 2013.
- [22] EUROCONTROL. Airport collaborative decision-making (A-CDM). URL <https://www.eurocontrol.int/concept/airport-collaborative-decision-making>. Accessed: 03-03-2021.
- [23] EUROCONTROL. European Aviation in 2040, Challenges of Growth. Technical report, EUROCONTROL, 2018.
- [24] EUROCONTROL. EUROCONTROL Specification for Advanced-Surface Movement Guidance and Control System (A-SMGCS) Services (EUROCONTROL-SPEC-171). Technical report, EUROCONTROL, 2020.
- [25] EUROCONTROL. EUROCONTROL Forecast Update 2021-2024 European Flight Movements and Service Units Three Scenarios for Recovery from COVID-19. Technical report, EUROCONTROL, 2021.
- [26] EUROCONTROL Airport CDM Team. Airport Collaborative Decision-Making (A-CDM) Implementation Manual Version 5.0. Technical report, EUROCONTROL, 2017.
- [27] 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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, 7 2012.
- [28] Patricia A Martin Fernandez, Radesh Nagaraj, Frank Speelman, and Xander Mobertz. Sustainable taxi at Schiphol Airport FTS Study. Technical report, TO70 Aviation Consultants, 2020.
- [29] Konstantine Fines. Agent-based distributed planning and coordination for resilient airport surface movement operations, 2019. Delft University of Technology.
- [30] Konstantine Fines, Alexei Sharpanskykh, and Matthieu Vert. Agent-Based Distributed Planning and Coordination for Resilient Airport Surface Movement Operations. *Aerospace*, 7(4):48, 4 2020. ISSN 2226-4310. doi: 10.3390/aerospace7040048.
- [31] Rui Guo, Yu Zhang, and Qing Wang. Comparison of emerging ground propulsion systems for electrified aircraft taxi operations. *Transportation Research Part C: Emerging Technologies*, 44:98–109, 7 2014. ISSN 0968090X. doi: 10.1016/j.trc.2014.03.006.
- [32] Lu Hao, Megan S. Ryerson, Lei Kang, and Mark Hansen. Estimating fuel burn impacts of taxi-out delay with implications for gate-hold benefits. *Transportation Research Part C: Emerging Technologies*, 80: 454–466, 7 2017. ISSN 0968090X. doi: 10.1016/j.trc.2016.05.015.
- [33] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. ISSN 21682887. doi: 10.1109/TSSC.1968.300136.
- [34] Wolfgang Hönig, T. K. Satish Kumar, Liron Cohen, Hang Ma, Hong Xu, Nora Ayanian, and Sven Koenig. Multi-Agent Path Finding with Kinematic Constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 477–485, 2016.

- [35] Wolfgang Honig, James A. Preiss, T. K. Satish Kumar, Gaurav S. Sukhatme, and Nora Ayanian. Trajectory Planning for Quadrotor Swarms. *IEEE Transactions on Robotics*, 34(4):856–869, 8 2018. ISSN 15523098. doi: 10.1109/TRO.2018.2853613.
- [36] Wolfgang Hönig, Scott Kiesel, Andrew Tinka, Joseph W. Durham, and Nora Ayanian. Persistent and Robust Execution of MAPF Schedules in Warehouses. *IEEE Robotics and Automation Letters*, 4(2):1125–1131, 2019. ISSN 23773766. doi: 10.1109/LRA.2019.2894217.
- [37] Jakub Hospodka. Cost-benefit analysis of electric taxi systems for aircraft. *Journal of Air Transport Management*, 39:81–88, 7 2014. ISSN 09696997. doi: 10.1016/j.jairtraman.2014.05.002.
- [38] IAI. Airplane taxi: TaxiBot performs Pushback operations and procedures. URL <https://www.iai.co.il/p/taxibot>. Accessed: 29-04-2021.
- [39] ICAO. Rules of the Air - Annex 2 to the Convention on International Civil Aviation International Civil Aviation Organization International Standards (Tenth Edition). Technical report, International Civil Aviation Organization, 7 2005.
- [40] ICAO. Annex 14 - Aerodromes - Aerodromes Design and Operations (Eight Edition). Technical report, International Civil Aviation Organization, 2018.
- [41] International Civil Aviation Organization. International Civil Aviation Organization Advanced Surface Movement Guidance and Control Systems (A-SMGCS) Manual (Doc 9830). Technical report, 2004.
- [42] 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.
- [43] M. Khorshid, R. Holte, and Nathan R Sturtevant. A Polynomial-Time Algorithm for Non-Optimal Multi-Agent Pathfinding. In *Proceedings of the 4th Annual Symposium on Combinatorial Search (SOCS)*, pages 76–83, 2012.
- [44] Sang Hyun Kim and Eric Feron. Impact of gate assignment on gate-holding departure control strategies. In *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, Williamsburg, VA, USA, 2012. IEEE. ISBN 9781467316996. doi: 10.1109/DASC.2012.6382350.
- [45] KLM. KLM trials sustainable taxiing, 5 2020. URL <https://news.klm.com/klm-trials-sustainable-taxiing/>. Accessed: 10-03-2021.
- [46] Jiaoyang Li, Pavel Surynek, Ariel Felner, Hang Ma, T. K. Satish Kumar, and Sven Koenig. Multi-agent path finding for large agents. In *33rd AAAI Conference on Artificial Intelligence (AAAI-2019)*, volume 33, pages 7627–7634. AAAI Press, 7 2019. ISBN 9781577358091. doi: 10.1609/aaai.v33i01.33017627.
- [47] Ryan Luna and Kostas E. Bekris. Push and swap: Fast cooperative path-finding with completeness guarantees. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 294–300, 2011. ISBN 9781577355120. doi: 10.5591/978-1-57735-516-8/IJCAI11-059.
- [48] LVNL. Baangebruik Schiphol, 2022. URL <https://www.lvn1.nl/omgeving/baangebruik>. Accessed on: 20-06-2022.
- [49] LVNL Aeronautical Information Services. EHAM - AMSTERDAM/Schiphol, 2021. URL <https://www.lvn1.nl/eaip/2021-02-11-AIRAC/html/index-en-GB.html>. Accessed: 02-03-2021.
- [50] Hang Ma and Sven Koenig. AI Buzzwords Explained: Multi-Agent Path Finding (MAPF). *AI Matters*, 3(3):15–19, 2017. doi: 10.1145/3137574.3137579.
- [51] Hang Ma, Sven Koenig, Nora Ayanian, Liron Cohen, Wolfgang Hönig, T K Satish Kumar, Tansel Uras, Hong Xu, Craig Tovey, and Guni Sharon. Overview: Generalizations of Multi-Agent Path Finding to Real-World Scenarios. In *Proceedings of the IJCAI-16 Workshop on Multi-Agent Path Finding*, 2016.
- [52] Hang Ma, Daniel Harabor, Peter J. Stuckey, Jiaoyang Li, and Sven Koenig. Searching with Consistent Prioritization for Multi-Agent Path Finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7643–7650, 2019. doi: <https://doi.org/10.1609/aaai.v33i01.33017643>.

- [53] Hang Ma, Wolfgang Hönig, T K Satish Kumar, Nora Ayanian, and Sven Koenig. Lifelong Path Planning with Kinematic Constraints for Multi-Agent Pickup and Delivery. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 7651–7658, 2019. doi: <https://doi.org/10.1609/aaai.v33i01.33017651>.
- [54] Robert Morris, Corina S Pasareanu, Kasper Luckow, Waqar Malik, Hang Ma, T K Satish Kumar, and Sven Koenig. Planning, Scheduling and Monitoring for Airport Surface Operations. In *The Workshops of the Thirtieth AAAI Conference on Artificial Intelligence Planning for Hybrid Systems.*, pages 608–614, 2016.
- [55] Timo Noortman. Agent-Based Modelling of an Airports Ground Surface Movement Operation: Understanding the principles and mechanisms of decentralised control, 2018. Delft University of Technology.
- [56] Nikolai Okuniek and Dustin Beckmann. Towards higher level of A-SMGCS: Handshake of electric taxi and trajectory-based taxi operations. In *IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*, pages 1–10. Institute of Electrical and Electronics Engineers Inc., 11 2017. ISBN 9781538603659. doi: 10.1109/DASC.2017.8102047.
- [57] OpenStreetMap Contributors. OpenStreetMap, 2021. URL <https://www.openstreetmap.org/copyright>. Accessed: 28-04-2021.
- [58] Mike Phillips and Maxim Likhachev. SIPP: Safe interval path planning for dynamic environments. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 5628–5635, Shanghai, China, 2011. IEEE. ISBN 9781612843865. doi: 10.1109/ICRA.2011.5980306.
- [59] Spyros Polydorou. A learning-based approach for distributed airport surface movement operations (Literature Study), 2020.
- [60] Stefan Ravizza, Jason A.D. Atkin, and Edmund K. Burke. A more realistic approach for airport ground movement optimisation with stand holding. *Journal of Scheduling*, 17(5):507–520, 9 2014. ISSN 10946136. doi: 10.1007/s10951-013-0323-3.
- [61] Craig W Reynolds. Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics*, 21(4), 7 1987.
- [62] Frédéric Rooseleer and Vincent Treve. "RECAT-EU" European Wake Turbulence Categorisation and Separation Minima on Approach and Departure Edition 1.2. Technical report, EUROCONTROL, feb 2018. URL www.eurocontrol.int.
- [63] Royal Schiphol Group. Sustaining Your World - Vision and strategy towards the most sustainable airports, 2020. URL <https://www.schiphol.nl/en/schiphol-group/page/road-to-the-most-sustainable-airports/>. Accessed: 29-04-2021.
- [64] Stuart Russel and Peter Norvig. *Artificial Intelligence A Modern Approach*. Prentice Hall, 3 edition, 2010. ISBN 978-0-13-604259-4.
- [65] Schiphol. Schiphol Tomorrow | Voltooing dubbele rijbaanstelsel. URL <https://www.schiphol.nl/nl/projecten/pagina/voltooiing-dubbele-rijbaanstelsel/>. Accessed: 03-03-2021.
- [66] Schiphol. Safety & Security Pocket Guide (Version April 2017). Technical report, Schiphol, 2017.
- [67] Schiphol. Schiphol met CDM aangesloten op Europees netwerk, 2018. URL <https://nieuws.schiphol.nl/schiphol-met-cdm-aangesloten-op-europees-netwerk/>. Accessed: 03-03-2021.
- [68] Schiphol. Traffic review 2019. Technical report, Schiphol, 2019.
- [69] Schiphol. Update: Schiphol zet volgende stap naar kern Schiphol en sluit pieren, 4 2020. URL <https://nieuws.schiphol.nl/schiphol-blijft-open-met-fors-kleinere-operatie/>. Accessed: 01-03-2021.
- [70] Schiphol. Bedrijfshandboek Amsterdam Airport Schiphol: Deel 1 Aerodrome Manual (version 7.0). Technical report, Schiphol, 2020.
- [71] Schiphol. RASAS: Regulation Aircraft Stand Allocation Schiphol. Technical report, Schiphol, 2020.

- [72] Schiphol. Handboek Landingsterrein: Deel 1.2.5 Uitvoeren Pushbackbewegingen (version 5.0). Technical report, Schiphol, 2020.
- [73] Schiphol. Schiphol | Airport history, 2021. URL <https://www.schiphol.nl/en/you-and-schiphol/page/airport-history/>. Accessed: 01-03-2021.
- [74] Schiphol. Schiphol | Nieuwe pier, 2021. URL <https://www.schiphol.nl/nl/projecten/pagina/project-nieuwe-pier/>. Accessed on: 01-03-2021.
- [75] Schiphol. Schiphol | Reis zoals je wilt reizen via Amsterdam Airport Schiphol, 2021. URL <https://www.schiphol.nl/nl/operations/pagina/sleep-en-pushbackbewegingen/>. Accessed on: 29-04-2021.
- [76] Schiphol. Schiphol | An autonomous airport in 2050, 2021. URL <https://www.schiphol.nl/en/innovation/page/an-autonomous-airport-in-2050/>. Accessed: 29-04-2021.
- [77] Schiphol Airport. Schiphol Airport CDM Operations Manual (Version 1.3). Technical report, Schiphol, 2019.
- [78] Schiphol Airport. Final results Taxibot Proof of concept. Technical report, Schiphol, 2021.
- [79] Schiphol Airport. Feasibility Study Sustainable Taxiing at Schiphol. Technical report, Schiphol, 2021.
- [80] Schiphol HSE Office/DVD. Schipholregels (version 29.0). Technical report, Schiphol, 2020.
- [81] SESAR. SESAR Joint Undertaking | Background on Single European Sky. URL <https://www.sesarju.eu/background-ses>. Accessed: 19-06-2021.
- [82] SESAR. Projects - A-SMGCS High Performance Surveillance enhancement in view to support routing & planning functions implementation | SESAR DM, 2017. URL https://www.sesardeploymentmanager.eu/projects/2017_063_af2. Accessed: 09-03-2021.
- [83] Guni Sharon, Roni Stern, Meir Goldenberg, and Ariel Felner. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*, 195:470–495, 2 2013. ISSN 00043702. doi: 10.1016/j.artint.2012.11.006.
- [84] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2 2015. ISSN 00043702. doi: 10.1016/j.artint.2014.11.006.
- [85] Alexei Sharpanskykh. Agent-based Modelling and Simulation in Air Transport lecture: Introduction to agents and multiagent systems. Emergence, 2020. Delft University of Technology.
- [86] Alexei Sharpanskykh. Agent-based Modelling and Simulation in Air Transport lecture: Multiagent Planning and Scheduling, 2020. Delft University of Technology.
- [87] Alexei Sharpanskykh. Agent-based Modelling and Simulation in Air Transport lecture: Specification of models of multiagent systems. Languages for multiagent systems, 2020. Delft University of Technology.
- [88] David Silver. Cooperative Pathfinding. In *Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2005.
- [89] Ioannis Simaiakis, Harshad Khadilkar, Hamsa Balakrishnan, Tom G. Reynolds, and R. John Hansman. Demonstration of reduced airport congestion through pushback rate control. *Transportation Research Part A: Policy and Practice*, 66(1):251–267, 8 2014. ISSN 09658564. doi: 10.1016/j.tr.2014.05.014.
- [90] Joost Soomers. Agent-based Delay Management in Autonomous Engine-off Taxi Systems (Literature study), 2021. Delft University of Technology.
- [91] Joost Soomers. A Delay Analysis in Airport Engine-off Towing Operations Using Hierarchical Multi-Agent Path Planning, jun 2022. Delft University of Technology.
- [92] Trevor Standley. Finding Optimal Solutions to Cooperative Pathfinding Problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 7 2010.

- [93] Roni Stern. Multi-agent path finding an overview. In G. Osipov, A. Panov, and K. Yakovlev, editors, *Artificial Intelligence, LNAI*, volume 11866, pages 96–115. Springer, 2019. ISBN 9783030332730. doi: 10.1007/978-3-030-33274-7_6.
- [94] Roni Stern, Nathan R Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T K Satish Kumar, Eli Boyarski, and Roman Barták. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *Proceedings of the Symposium on Combinatorial Search (SoCS)*, pages 151–158, 2019.
- [95] Pavel Surynek. Towards optimal cooperative path planning in hard setups through satisfiability solving. In *Pacific Rim International Conference on Artificial Intelligence (PRICAI)*, pages 564–576, 2012.
- [96] Smart Airport Systems. TaxiBot® - Smart Airport System. URL <https://www.smart-airport-systems.com/solutions/taxibot/>. Accessed: 13-06-2021.
- [97] Heiko Udluft. *Decentralization in Air Transportation*. PhD thesis, Delft University of Technology, Delft, 12 2017.
- [98] Edzard van Baaren and Paul C Roling. Design of a zero emission aircraft towing system. In *AIAA Aviation 2019 Forum*, pages 2019–2932. American Institute of Aeronautics and Astronautics Inc, AIAA, 2019. ISBN 9781624105890. doi: 10.2514/6.2019-2932. URL <https://doi.org/10.2514/6.2019-2932>.
- [99] Vanderlande. Rotterdam The Hague Airport | Vanderlande. URL <https://www.vanderlande.com/references/rotterdam-the-hague-airport/>. Accessed: 24-03-2021.
- [100] András Vargha and Harold D. Delaney. A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics*, 25(2):132, 2000. ISSN 10769986. doi: 10.2307/1165329.
- [101] Chris Wijnterp, Paul C Roling, Wido De Wilde, and Richard Curran. Electric Taxi Systems: An operations and value estimation. In *14th AIAA Aviation Technology, Integration, and Operations Conference*, Atlanta, GA, USA, 6 2014. doi: 10.2514/6.2014-3266.
- [102] Jingjin Yu and Steven M. Lavalle. Planning optimal paths for multiple robots on graphs. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3612–3617, 2013. ISBN 9781467356411. doi: 10.1109/ICRA.2013.6631084.